APOSTILA DE PROGRAMAÇÃO EM VBA

Profª Drª Mariana Kleina

Universidade Federal do Paraná

Curitiba 2022

Apostila elabora para ser usada na disciplina *Informática para Engenharia de Produção* da UFPR

Sumário

1. CONSIDERAÇÕES INICIAIS

1.1 Habilitar Macro no Excel

Excel 2007: Botão do Microsoft Office → Opções do Excel → Central de Confiabilidade → Configurações da Central de Confiabilidade → Configurações de Macro → Habilitar todas as Macros.

Excel 2013 em diante: Arquivo → Opções → Central de Confiabilidade → Configurações da Central de Confiabilidade → Configurações de Macro → Habilitar todas as Macros.

1.2 Habilitar guia Desenvolvedor

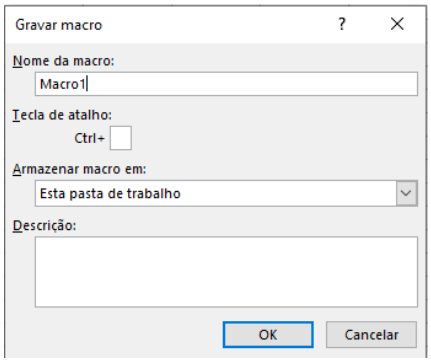
Excel 2007: Botão do Microsoft Office → Opções do Excel → Mais Usados → Mostrar guia Desenvolvedor na Faixa de Opções.

Excel 2013 em diante: Arquivo → Opções → Personalizar Faixa de Opções → Em Guias Principais, marcar a caixa de seleção Desenvolvedor.

1.3 Gravar uma Macro

Desenvolvedor → Gravar Macro (digitar nome da macro, e clicar em ok). Digitar ações que deseja gravar → Parar Gravação.

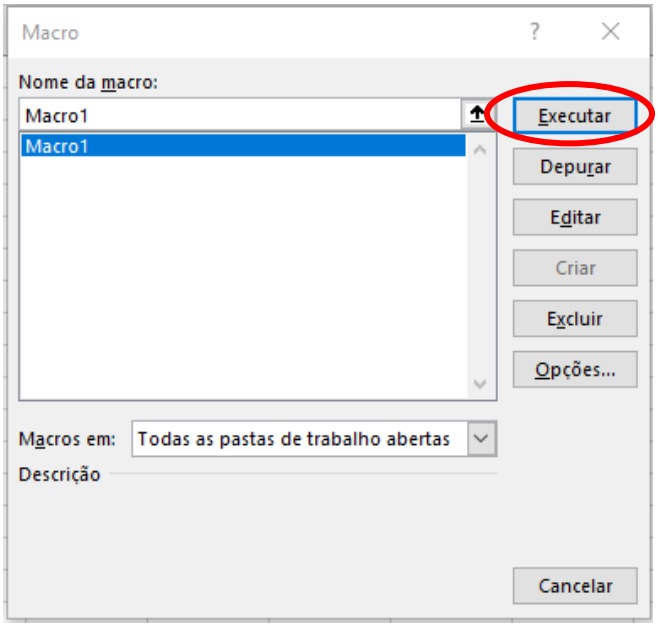
Além do nome da Macro, é possível atribuir um atalho no teclado para depois executá-la, e também uma descrição do que ela faz.



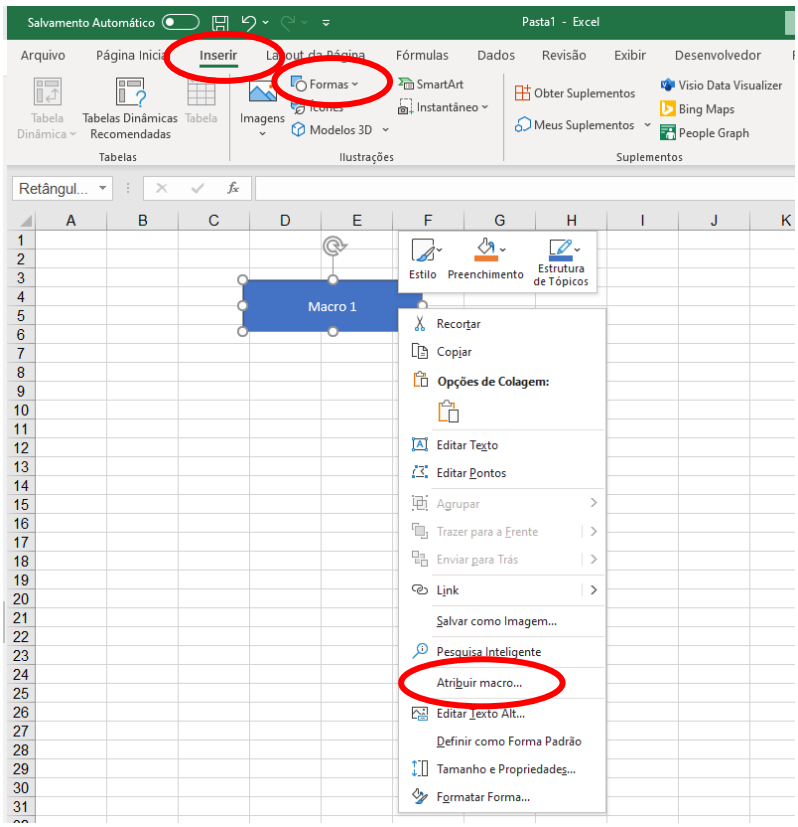
Após a gravação dos comandos que determinada macro irá realizar, é preciso encerrar a gravação.

1.4 Executar uma Macro

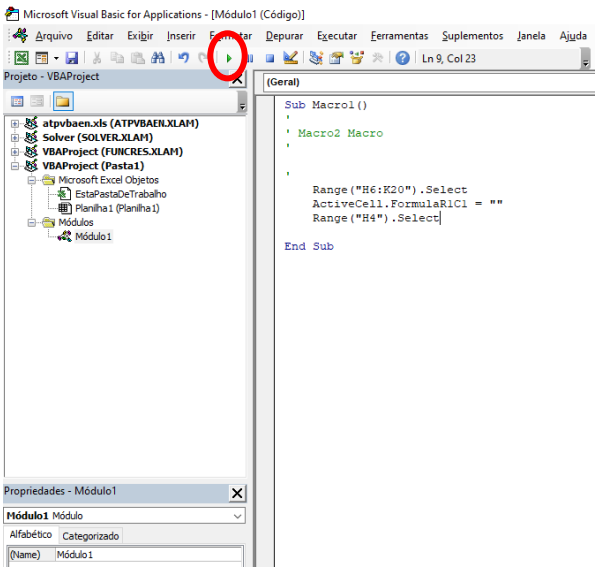
Maneira 1: Desenvolvedor → Macros → Escolher a macro e clicar em Executar.



Maneira 2: Inserir uma forma, editar o texto da forma, clicar com o botão direito nela e selecionar "Atribuir macro".

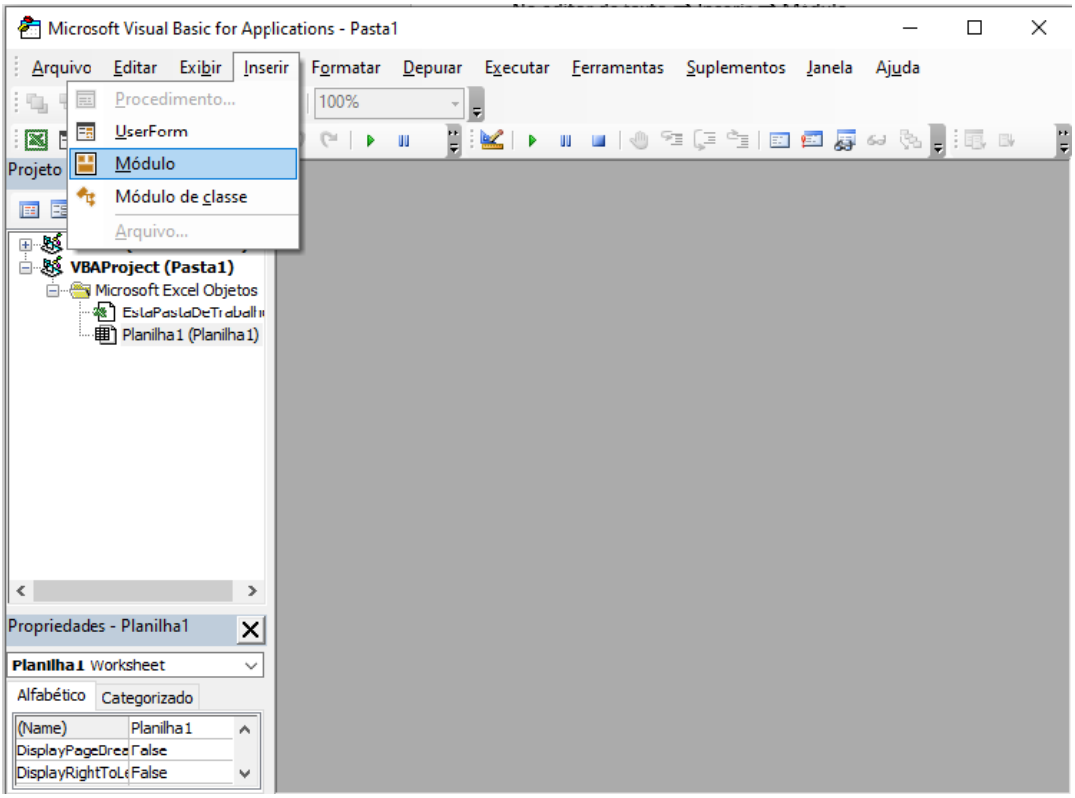


Maneira 3: Entrar no editor de texto do VBA (Alt+F11 ou Desenvolvedor → Visual Basic), deixar o cursor do mouse em qualquer lugar da macro e clicar em Executar (F5 ou ▶).



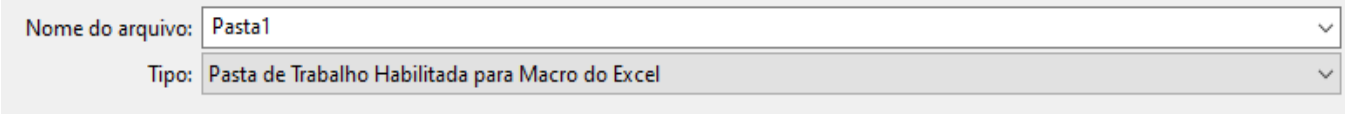
1.5 Escrever suas próprias macros

No editor de texto → Inserir → Módulo.



1.6 Salvando a planilha do Excel

Para salvar a planilha Excel e não perder as informações dos códigos gravados e/ou escritos, é preciso salvar o arquivo como **Pasta de Trabalho Habilitada para Macro do Excel**. Salvando dessa forma, o arquivo terá a extensão .xlsm.



2. VARIÁVEIS E CONSTANTES

2.1 Variáveis

Para declarar uma variável no VBA, usa-se:

```
Dim nome_da_variavel As Tipo
```

Onde `Tipo` pode ser:

- `Integer`: números inteiros (de -32768 a 32767)
- `Long`: inteiros mais longos (de -2147483648 a 2147483647)
- `Single`: número real de precisão simples (7 algarismos)
- `Double`: número real de precisão dupla (15 algarismos)
- `Boolean`: valores lógicos `True` (verdadeiro) ou `False` (falso)
- `String`: texto (caracteres) e deve ser escrito entre aspas dupla
- `Variant`: válido para qualquer tipo de dados

Obs 1: `Option Explicit` na primeira linha do editor de texto obriga a declaração de todas as variáveis (sempre recomendado usar). É possível configurar o editor de texto para que toda vez que um novo módulo é criado, o termo `Option Explicit` já venha escrito (no editor do VBA clique em: Ferramentas → Opções → Editor → e marque a caixa de seleção "Requer declaração de variáveis").

Obs 2: O tipo `Variant` é usado quando uma variável pode receber diversos tipos ao longo do código. Por exemplo, uma variável pode ser inteira ou string ao longo de um programa, logo ela precisa ser declarada com `Variant`. Mas só use `Variant` quando realmente for necessário, pois ocupa mais espaço na memória do computador do que outras variáveis.

Obs 3: Nomes de variáveis sempre devem começar por uma letra e as demais devem ser letras, números e/ou underline. Não são permitidos espaços ou caracteres especiais tais como: !@#$%&*()+-/<>?.,;{}[]. Não são permitidas palavras reservadas (`As`, `Dim`, `For`, `If`, `While`,...), e nessa apostila, assim como no VBA, palavras reservadas estarão destacadas na cor azul.

Obs 4: O VBA não faz distinção entre letra maiúscula e minúscula.

Obs 5: No VBA, usa-se ponto (e não a vírgula) para separar a parte inteira da parte decimal em números reais (`Single` ou `Double`).

Obs 6: Utiliza-se o apostrofo (') para indicar um comentário no código. Comentários ficarão em verde nesta apostila, assim como no VBA, e sempre são ignorados durão a execução do código. É interessante a utilização do comentário para descrever o que cada parte do código faz, principalmente quando o código é mais longo ou quando uma pessoa irá olhar um código não escrito por ela.

2.2 Constantes

Para declarar uma constante, usa-se:

```
Const nome_da_constante As Tipo = valor
```

O `Tipo` é mesmo usado em variáveis, e `valor` é um valor que já deve ser informado, indicando o conteúdo da constante. Uma constante não pode ter seu valor alterado durante o programa.

2.3 Funções de entrada e saída de dados

São funções que permitem que o usuário insira ou receba dados para utilizar em seu código. Tais dados podem ser atribuídos a variáveis ou variáveis podem ser impressas para o usuário visualizar, por exemplo.

O comando `InputBox` permite que o usuário entre com uma informação e esta deve ser armazenada em uma variável. Verificar se o Excel que você está usando exige que números reais sejam digitados com ponto ou vírgula (varia de versão) quando usado `InputBox`.

```
a = InputBox("Insira uma mensagem solicitando a informação desejada")
```

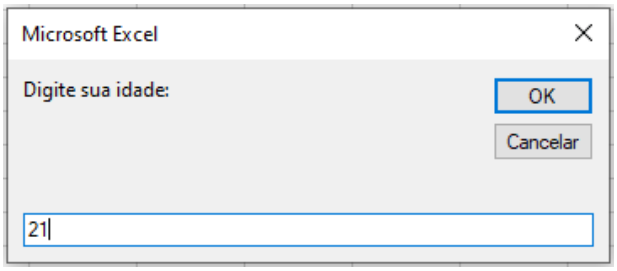
Já o comando `MsgBox` mostra o conteúdo de uma variável em uma tela. No exemplo a seguir, `valor` é impresso na tela.

```
MsgBox(valor)
```

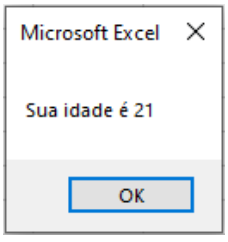
Exemplo:

```
Sub Idade()
     Dim idade As Integer
     idade = InputBox("Digite sua idade: ")
     MsgBox("Sua idade é " & idade)
     ' o símbolo & é utilizado para concatenar variáveis e texto
End Sub
```

O resultado da execução da subrotina `Idade` será:



Portanto, será aberta uma janela solicitando que o usuário digite algo do teclado. Após digitar e apertar OK, uma nova janela será aberta.

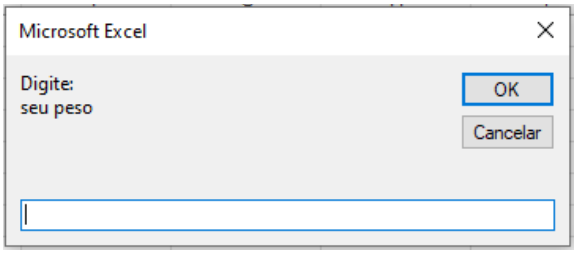


Uma dica importante quando se usam as funções `InputBox` e `MsgBox` é quando se quer quebrar linha na mensagem da caixa de texto. Para isso usa-se o comando `vbNewLine`, o qual deve ser usado entre o símbolo `&`.

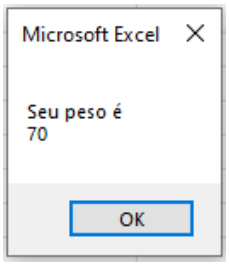
Exemplo:

```
Sub Peso()
     Dim peso As Single
     peso = InputBox("Digite: " & vbNewLine & "seu peso")
     MsgBox("Seu peso é " & vbNewLine & peso)
End Sub
```

O resultado será:



E após inserido o valor 70, por exemplo, tem-se:



Uma outra maneira de fornecer ou imprimir informações diretamente do Excel é por meio dos comandos `Cells(linha,coluna)` e `Range("Celula_do_Excel")`.

Exemplos:

`a = Cells(3,5)` → a variável `a` está recebendo o conteúdo da célula que está na linha 3 e coluna 5 do Excel (célula C5).

`Cells(4,2) = 100` → a célula correspondente a linha 4 e coluna 2 (célula D2) está recebendo o valor 100.

`Range("B7") = 2` → a célula B7 está recebendo o valor 2.

2.4 Ativação de planilhas do Excel

Se comandos `Cells` ou `Range` são utilizados em um programa em VBA, por padrão a planilha aberta no seu Excel está ativada, e tudo é impresso nela ou lido diretamente dela. Porém é possível ativar outras planilhas do Excel da seguinte forma:

```
Sheets(*).Activate
```

* pode ser o nome da planilha (o nome deve estar entre aspas duplas) ou o número da planilha.

Exemplo:

```
Sub AtivandoPlanilha()
     Dim x As Single
     Sheets("Planilha2").Activate 'ou Sheets(2).Activate
     Cells(1,1) = "teste"
     x = Range("B1")
End Sub
```

Neste exemplo, a planilha chamada `Planilha2` foi ativada, e é feita a impressão de uma string na célula A1 e a leitura de um valor que está na célula B1. Tudo o que é feito abaixo do comando que ativa uma planilha, irá fazer referência a ela. Mas só é possível ativar uma planilha se ela já foi criada no Excel.

É possível ativar só uma célula de uma planilha, sem ativar toda a planilha.

Exemplo:

```
Sub AtivandoCelula()
     Sheets("Planilha2").Cells(1,1) = 5
     Cells(1,1)=10
End Sub
```

Neste exemplo, suponha que a primeira planilha do Excel esteja ativada antes da subrotina ser executada. Então a célula A1 da segunda planilha receberá o valor 5 e a célula A1 da primeira planilha receberá o valor 10.

3. OPERADORES

3.1 Operador de atribuição

Atribui um valor à uma variável, por meio do sinal =

Anteriormente já foram vistos vários exemplos utilizando o operador de atribuição quando se queria dar valor a uma variável, por exemplo.

Exemplos:

```
a = InputBox("Insira um valor inteiro")
b = Cells(3,5)
c = 500
```

3.2 Operadores aritméticos

Usados para realizar cálculos matemáticos. São eles:

`+` (adição)
`−` (subtração)
`*` (multiplicação)
`/` (divisão)
`\` (divisão inteira)
`Mod` (resto da divisão)
`^` (exponenciação)

Exemplos:

Se `a = 2` e `b = 3`, então é possível realizar, por exemplo, as operações:

```
c = a + b 'c vale 5
d = a - b 'd vale -1
e = a * b 'e vale 6
f = a / b 'f vale 0,6666667
g = a \ b 'g vale 0
h = a Mod b 'h vale 2
i = a ^ b 'i vale 8
```

Algumas funções matemáticas prontas no VBA:

`Sqr(n)` retorna a raiz quadrada de um número positivo `n`;
`Exp(n)` retorna o número e (logaritmo neperiano $e = 2{,}7183$) elevado ao número `n`;
`Log(n)` retorna o logaritmo natural do número `n`;
`Abs(n)` retorna o valor absoluto de um número real `n`;
`Sgn(n)` retorna -1 se `n` é negativo e 1 se `n` é positivo;
`Rnd()` retorna um valor aleatório real entre 0 e 1.

Funções trigonométricas no VBA, onde `x` é dado em radianos:

`Sin(x)` retorna o seno de `x`;
`Cos(x)` retorna o cosseno de `x`;
`Tan(x)` retorna a tangente de `x`;
`Atn(x)` retorna o arco cuja tangente é `x`.

Para se arredondar números reais para valores inteiros, existem algumas formas:

- Dimensionar uma variável inteira, e ao se atribuir um valor real decimal o valor é automaticamente arredondado para cima se a parte decimal for maior ou igual a 5, e para baixo se a parte decimal é menor que 5;
- A função `Round(x,n)` arredonda o valor real `x` com `n` casas após a vírgula;
- A função `Int(x)` arredonda um número real para baixo, ou seja, retorna a parte inteira do número `x`;
- A função `Application.WorksheetFunction.RoundUp(x, 0)`, arredonda um número real `x` para cima.
  Obs: A expressão `Application.WorksheetFunction` significa que uma função pronta do Excel (e não do VBA) será usada (no caso a função do Excel se chama `RoundUp`). A nossa própria função de arredondamento para cima pode ser criada no VBA, porém necessita de conceitos de manipulação de strings.
  Lista de funções do Excel que podem ser chamadas usando o objeto `WorkSheetFunction`: https://docs.microsoft.com/pt-br/office/vba/excel/concepts/events-worksheetfunctions-shapes/list-of-worksheet-functions-available-to-visual-basic

3.3 Operadores de comparação

Operação	Igual a	Diferente de	Maior que	Maior ou igual a	Menor que	Menor ou igual a
Operador	=	<>	>	>=	<	<=

São utilizados para comparar variáveis e/ou expressões e retornam um valor booleando (`True` ou `False`).

Exemplo:

```
Dim a As Integer, b As Integer, c As Boolean

a = 2
b = 3

c = a = b 'c vale False
c = a <> b 'c vale True
c = a > b 'c vale False
c = a >= b 'c vale False
c = a < b 'c vale True
c = a <= b 'c vale True
```

3.4 Operadores lógicos

Operação	**Operador**	**Expressão usual**
Conjunção	`And`	e
Disjunção	`Or`	ou

Negação	Not	não

O resultado da utilização de operadores lógicos é booleano e segue a regra:

Conjunção	Disjunção	Negação
True And True = True	True Or True = True	Not True = False
True And False = False	True Or False = True	Not False = True
False And True = False	False Or True = True	
False And False = False	False Or False = False	

3.5 Precedência de operadores

Em que ordem as operações são realizadas no VBA?

Aritméticos: ^, (*, /), \, Mod, (+,–)

Comparação: avaliados da esquerda para a direita na ordem em que aparecem
Lógicos: Not, And e Or

Obs: parênteses priorizam operações.

4. SUBROTINAS E FUNÇÕES

Funções são muito similares às subrotinas em relação ao seu funcionamento, mas não às suas funcionalidades e utilidades. As principais diferenças são:

- Subrotinas não retornam valores, se chamadas. Funções retornam apenas valores.
- Subrotinas sem parâmetros podem ser associadas à botões e formas do Excel, funções não.
- Funções criadas no VBA podem ser utilizadas na sua planilha em Excel.
- Funções necessitam de subrotinas que as chamem para que sejam executadas no VBA.
- O nome de uma função sempre deve receber a variável de retorno da função.

Antes de exemplificar a diferença entre subrotina e função, é necessário entender o que é um parâmetro.

4.1 Parâmetros

Parâmetros ou argumentos são valores que devem ser fornecidos à subrotinas e/ou funções para que elas tenham sua execução diferenciada, conforme o valor que lhe é fornecido. Por exemplo, dia, mês e ano devem ser parâmetros para uma função ou subrotina que calcula a idade de uma pessoa. Sem estas informações é impossível fazer o cálculo. Veja que datas de nascimento diferentes geram resultados diferentes.

Uma subrotina ou função que calcula o índice de massa corporal de uma pessoa, necessita de dois parâmetros: o peso e a altura. É importante destacar que se o programa que calcula o índice de massa corporal foi escrito de tal forma que peso é o primeiro parâmetro e altura o segundo, então no momento de se chamar o programa, devem ser fornecidos peso e altura (nesta ordem obrigatoriamente).
Algumas subrotinas e funções não necessitam de parâmetro(s), como por exemplo uma função que simule o sorteio de um lançamento de um dado. Veja que não é preciso fornecer nenhuma informação para se lançar um dado.

4.2 Subrotinas sem parâmetros

```
Sub nome_da_subrotina()
      .
      .
      .
End Sub
```

Nesse caso, a subrotina é capaz de ser executada independente, sem a necessidade de fornecer informações externas ou as informações são fornecidas dentro da própria subrotina (por exemplo, um `InputBox` para ler um valor do teclado).
Subrotinas sem parâmetros também podem ser chamadas dentro de outras subrotinas. Para isso, apenas escreve-se o nome da subrotina onde se quer chamá-la.

4.3 Subrotinas com parâmetros

Uma subrotina com parâmetros é uma subrotina que pode ser chamada dentro de outra subrotina a fim de executar uma tarefa específica com os dados parametrizados disponibilizados.

```
Sub nome_da_subrotina(p1 As Tipo, p2 As Tipo,...)
      .
      .
      .
End Sub
```

Subrotinas com parâmetros não rodam sozinhas. Deve-se criar uma outra subrotina sem parâmetros para chamá-las.

Obs: Quando é feita a chamada de uma subrotina que contém parâmetros, estes devem ser fornecidos **na ordem** com que foram definidos.

Subrotinas com parâmetros são mais comuns quando se trabalha com vetores e matrizes, os quais são passados vazios como parâmetros e são preenchidos (tamanho e elementos) ou quando se deseja modificar eles em uma subrotina específica e usá-los em outro programa. Mas vamos a um exemplo com variáveis simples.

Exemplo: Calcular a média de idade de uma turma de 5 alunos. Suas idades estão preenchidas em uma planilha começando na célula B2 até a célula B6 conforme a tabela a seguir.

	A	B
1	**NOME**	**IDADE**
2	Caio	23
3	Felipe	25
4	José	33
5	Matheus	29
6	Carlos	26
7		

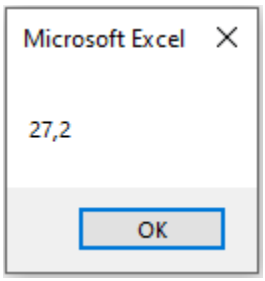
A subrotina `media` realiza o cálculo da média e apresenta o resultado por meio de uma caixa de mensagens.

```
Sub media(soma As Single, n As Single)
     Dim media As Single
     media = soma / n
     MsgBox (media)
End Sub
```

Já a subrotina `chama` é utilizada para obter os dados necessários para o cálculo e chamar a subrotina que calcula a média.

```
Sub chama()
     Dim soma As Single
     soma = Range("B2") + Range("B3") + Range("B4") + Range("B5") + _
     Range("B6")
     media soma, 5
End Sub
```

Tem-se como resultado então a média das idades fornecidas no intervalo de células B2 à B6.



Para chamar uma subrotina com parâmetros é preciso fornecer primeiramente o nome da subrotina e depois os seus parâmetros separados por vírgula, exatamente na ordem que foram definidos no momento da criação da subrotina.

Obs: Na subrotina `chama`, na linha onde é calculada a variável `soma`, foi inserido o caractere *underline* (_), com um espaço antes dele, para indicar que há uma quebra na linha de código (geralmente usado quando a linha é muito extensa). O *underline* pode ser usado em linhas de comandos, comentários, mensagens de `InputBox` e `MsgBox`,...).

4.4 Funções

Funções se diferenciam de subrotinas especialmente porque conseguem retornar um valor, e esse valor pode ser atribuído a uma variável em outro programa, por exemplo.

Da mesma forma como subrotinas, as funções podem ou não ter parâmetros. Será visto apenas o caso onde as funções tem parâmetros.

```
Function nome_da_funcao(p1 As Tipo, p2 As Tipo,...) As Tipo
     .
     .
     .
     nome_da_funcao = ...
End Function
```

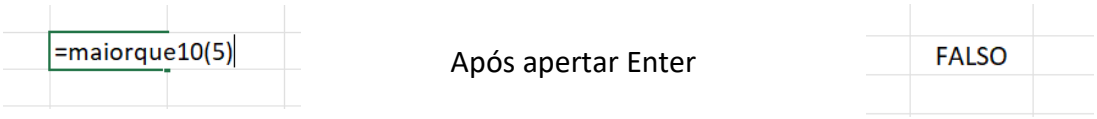
Observe que o valor que se deseja retornar deve obrigatoriamente ser atribuído a uma variável (que não pode ser declarada) com o mesmo nome da função. Essa variável de retorno da função tem um tipo associado a ela, e esse tipo é definido pelo programador (`As tipo` após o parênteses do cabeçalho da função).

Uma função deve ser chamada por uma subrotina no VBA:

```
Function maiorque10(x As Integer) As Boolean
     maiorque10 = x > 10
End Function

Sub chama_funcao()
     Dim resultado As Boolean 'este tipo deve ser o mesmo tipo do _
     retorno da função
     resultado = maiorque10(50)
     MsgBox(resultado)
End Sub
```

Observe que para chamar uma função ela deve ser atribuída a uma variável. Além disso, se a função tiver parâmetros, eles devem ser fornecidos entre vírgulas e entre parênteses após o nome da função.

Ou a função `maiorque10` pode ser chamada diretamente no Excel como uma fórmula:



4.5 Tipos de passagem de parâmetro

No VBA, há duas formas de se passar um parâmetro para uma subrotina ou função: por valor (`ByVal`) e por referência (`ByRef`).

- `ByVal`: é passado apenas uma cópia do valor da variável. Os comandos executados dentro da subrotina ou função não irão alterar o valor original da variável;
- `ByRef`: o endereço de memória da variável é passado, assim as alterações feitas na variável irão alterar o valor dela.

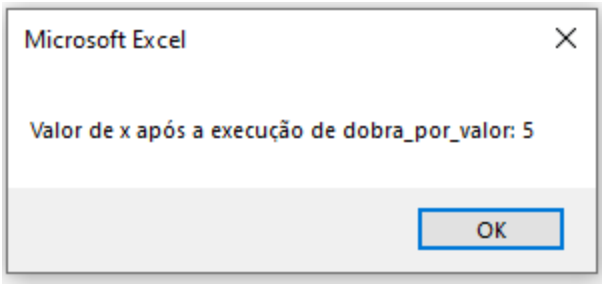
Caso a forma de passagem de parâmetro não seja informada, o VBA considera por padrão que os parâmetros são passados de modo `ByRef`.

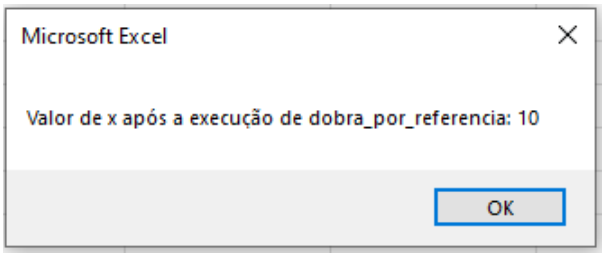
As palavras `ByRef` e `ByVal` aparecem antes do nome do parâmetro entre parênteses no cabeçalho da subrotina oufunção.

Exemplo: Criar duas subrotinas, uma com passagem de parâmetro por valor e outra por referência, onde o parâmetro deve ter seu valor dobrado. Criar outra subrotina para mostrar os resultados de ambas.

```
Sub dobra_por_referencia(ByRef n As Integer) 'a palavra ByRef é opcional
     n = n * 2
End Sub


Sub dobra_por_valor(ByVal n As Integer)
     n = n * 2
End Sub


Sub testa_dobra()
     Dim x As Integer 'x é o nome da variável nesta subrotina, _
     apesar de ter sido chamado de n nas duas anteriores
     x = 5

     dobra_por_valor x
     Msgbox("Valor de x após a execução de dobra_por_valor: " & x)

     dobra_por_referencia x
     Msgbox("Valor de x após a execução de dobra_por_referencia: " & x)
End Sub
```

O resultado da execução da subrotina `testa_dobra` é a seguinte:





A primeira figura é referente a chamada da subrotina que passou o parâmetro por valor, ou seja, não alterou o valor original do parâmetro. Já a segunda figura é decorrente da chamada da subrotina que passou o parâmetro por referência, ou seja, o parâmetro saiu modificado da subrotina `dobra_por_referencia`.

Obs: Quando se está trabalhando com vetor ou matriz como parâmetro, o VBA só aceita o tipo de passagem `ByRef`.

4.6 Nomes repetidos de subrotinas e funções

Subrotinas ou funções não podem ter nomes repetidos em um mesmo módulo. Isso porque se duas subrotinas de um mesmo módulo se chamam `Teste`, por exemplo, e uma outra subrotina faz a chamada de `Teste`, o VBA não sabe qual das duas subrotinas executar.

Subrotinas ou funções podem ter nomes repetidos em módulos diferentes, entretanto é preciso ter cuidado. Se uma subrotina faz a chamada de outra subrotina, o VBA irá procurar no módulo corrente o nome da subrotina que se está chamando: se a subrotina é encontrada (e ela será a única com o aquele nome no módulo atual) então ela é executada; se a subrotina chamada não é encontrada no módulo atual, então o VBA irá procurar nos demais módulos (neste caso se ele encontrar mais do que uma subrotina com aquele nome nos outros módulos, ele não será capaz de executar). Por este motivo, é recomendado não repetir nome de subrotinas ou funções mesmo em módulos diferentes.

5. ESTRUTURAS CONDICIONAIS

Usadas para selecionar um conjunto de ações dependendo de certas condições.

```
If condição Then
     Ações
End If
```

No código anterior, condição é um valor booleano e ações são comandos que somente são executados se condição é verdadeira.

```
If condição Then
     ações1
Else
     ações2
End If
```

No código anterior, condição é um valor booleano. Se condição é verdadeira, então ações1 são executadas; se condição é falsa então ações2 são executadas.

```
If condição1 Then
     ações1
ElseIf condição2 Then
     ações2
End If
```

No código anterior, condição1 e condição2 são valores booleanos. Se condição1 é verdadeira, então ações1 são executadas e ações2 não são executadas; se condição1 é falsa e condição2 é verdadeira então ações2 são executadas.

```
If condição1 Then
     ações1
ElseIf condição2 Then
     ações2
Else
     açoes3
End If
```

No código anterior, condição1 e condição2 são valores booleanos. Se alguma das condições (ou a primeira ou a segunda) é verdadeira, então sua respectiva ação é executada. Se as duas forem falsas, então ações3 são executadas.

Obs: Podem existir quantos `ElseIf` forem necessários.

```
If condição1 Then
     ações1
ElseIf condição2 Then
     ações2
ElseIf condição3 Then
     ações3
ElseIf condição4 Then
     ações4
...
Else
     açoesn
End If
```

Exemplo:

```
Sub exemplo_condicional()
     Dim a As Integer, b As Integer, c As Single
     a = 2
     b = 3
     If a > b Then
          c = a + b
     Elseif a = b Then
          c = a * b
     Else
          c = a / b
     End If
End Sub
```

Nesse caso `a` não é maior do que `b`, então a segunda condição é analisada, onde nota-se que `a` também não é igual a `b`, restando apenas a última condição (só resta o caso de `a` ser menor do que `b`), que implica na ação `c = a/b`, portanto tem-se como resultado `c = 2/3`.

Não é obrigação um `If` terminar com um `Else`, porém é sempre aconselhável, pois ele garante que se nenhuma condição testada anteriormente seja verdadeira ainda assim alguma ação será executada, se for de interesse (ele funciona como uma espécie de canal de fuga).

6. ESTRUTURAS DE REPETIÇÃO

Utilizadas para repetir um certo conjunto de ações. Iterar, como comumente se chama, consiste em realizar ações repetidas vezes e de forma sucessiva.

6.1 For / Next

Nesta estrutura, define-se uma variável de incremento, também conhecida como iterador ou contador, que começa com um valor definido `n` e se incrementa em 1 unidade a cada repetição até chegar ao valor máximo definido `m`. Para esta variável normalmente utilizam-se as letras `i`, `j` ou `k`.

```
For i = n To m
     ações
Next
```

Chegando ao valor de `m`, `ações` ainda são executadas e `i` assume o valor `m+1`, momento em que a repetição acaba.
Nessa estrutura obviamente `m` é maior ou igual a `n`.

O incremento de 1 (que é o padrão) pode ser alterado utilizando-se a forma:

```
For i = n To m Step x
     ações
Next
```

A variável `i` começa com `n` e é incrementada em `x` unidades a cada repetição, até chegar ao valor de `m`. O `Step` pode ser negativo, desde que `n` seja maior ou igual a `m`.

Exemplo: Será exemplificado o uso das estruturas de repetição com o exemplo do cálculo da média das idades, já utilizado anteriormente. Pode-se usar o `For` para calcular a soma dos valores, de modo que ele percorra a planilha somando cada célula de forma iterativa.

```
Sub media_com_for()
     Dim i As Integer, soma As Single
     For i = 2 To 6
          soma = soma + Cells(i , 2)
     Next
End Sub
```

6.2 While / Wend

Essa estrutura é usada para repetir ações não um número fixo de vezes como é no For, mas sim enquanto uma condição é atendida, não se sabendo o número de vezes que a repetição irá ocorrer.

```
While condição
     ações
Wend
```

O conjunto de ações será repetida até que a condição de torne falsa. Sendo assim a condição deve iniciar verdadeira e se tornar falsa em algum momento (caso contrário, se criará um laço infinito).

Neste ponto vale frisar a importância de salvar o código antes de executá-lo. Se um laço infinito é criado e o código não foi salvo antes de executá-lo, as alterações realizadas podem ser perdidas. Para não perder um código quando se entra em um laço infinito, basta apertar *Esc* ou *Ctrl+Break*.

6.3 Do / Loop

Uma outra estrutura utilizada para realizar repetições é a Do/loop. Porém uma condição deve ser testada no meio das ações para realizar a saída do laço.

```
Do
     ações
     If condição Then
          Exit Do
     End If
Loop
```

Essa estrutura deve obrigatoriamente conter uma estrutura condicional para ser capaz de sair do laço em algum momento. As ações são repetidas até que a condição (que é testada a cada repetição) seja verdadeira e execute o comando para saída do laço (Exit Do). A estrutura condicional pode aparecer em qualquer lugar entre o Do e o Loop, ou seja, estar antes das ações (que tem o mesmo efeito da estrutura While/Wend), estar no meio das ações ou no final das ações. Só enfatizando que quando o comando Exit Do é executado, nada mais depois dele dentro do laço é executado e acaba-se a estrutura de repetição.

Obs: É recomendado **indentar** o código dentro de uma estrutura condicional ou de repetição.

Exemplo: Utilizando-se das estruturas While/Wend ou Do/Loop, pode-se ir além em nosso problema de média e generalizá-lo sem limitar a média à um número específico de células, ou seja, se forem colocados 5, 10, 2 ou 100 alunos na planilha, é possível calcular a média de todos esses alunos sem saber quantos alunos existem.
Uma maneira de resolver o problema é impor a condição de somar as células até que seja encontrada uma célula vazia, sinalizando o fim da tabela.

```
Sub media_While()
     Dim i As Integer, soma As Single, media As Single
     i = 2
     While Cells(i, 2) <> ""
         soma = soma + Cells(i, 2)
         i = i + 1
     Wend
     media = soma / (i - 2)
     'aqui a soma foi dividida por i-2 pois i já começou em 2, _
     ou seja, uma unidade a mais e também foi incrementada em uma _
     unidade a mais do que deveria dentro do While ao final dele
End Sub
```

Ou equivalente:

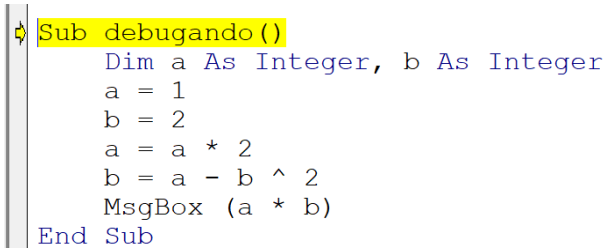
```
Sub media_Do()
     Dim i As Integer, soma As Single, media As Single
     i = 2
     Do
         soma = soma + Cells(i, 2)
         i = i + 1
         If Cells(i, 2) = "" Then
             Exit Do
         End If
     Loop
     media = soma / (i - 2)
End Sub
```

Obs: Assim como o comando `Exit Do` sai do laço, existem os comandos: `Exit For` (sai do laço do `For`), `Exit Function` (acaba a execução de uma função) e `Exit Sub` (finaliza a execução de uma subrotina).

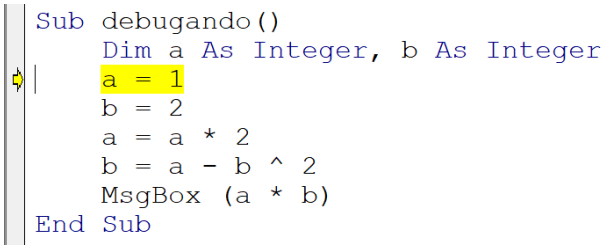
7. DEBUGAR OU DEPURAR UM CÓDIGO

O debug de um código tem a funcionalidade de identificar e remover qualquer tipo de erro de um código. Com a análise e execução linha a linha do programa, é possível encontrar erros e problemas como funções mal estruturadas ou erros de lógica de programação.

Para executar um código em VBA linha a linha e com o cursor do mouse dentro do código, deve-se apertar a tecla F8. A linha que é a próxima a ser executada fica destacada em amarelo (com uma seta na barra lateral esquerda), como na figura a seguir.

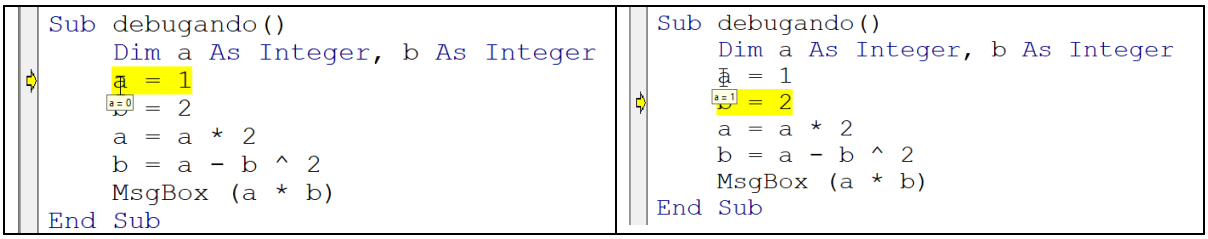


Para que a linha em amarelo seja executada e pule para a próxima linha, basta apertar novamente F8.



Obs: Linhas que contém dimensionamento de variáveis não ficam em amarelo, mas foram executadas normalmente.

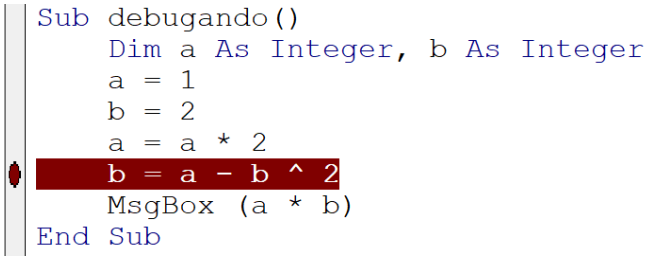
Lembrando que uma linha em amarelo para ser executada precisa que a tecla F8 seja apertada. Veja as figuras a seguir:



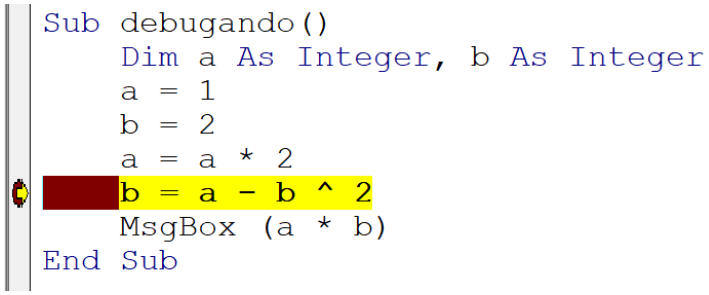
Na primeira figura, a linha `a = 1` ainda não foi executada, por isso quando o cursor do mouse fica em cima da variável `a`, aparece o valor 0. Já na segunda figura, a linha `a = 1` foi executada e a variável `a` tem o valor 1. Portanto, o valor de uma variável pode ser obtido deixando o cursor do mouse em cima da variável durante a execução do programa. Outra forma de verificar o valor de uma variável é clicando em cima da variável desejada com o botão direito do mouse e "Adicionar inspeção de variáveis" a ela. Com isso, uma caixa inferior na tela irá ficar disponível para visualizar o valor da variável conforme o código vai sendo executado.

Além disso, é possível adicionar *break points* (pontos de parada) no código, clicando com o botão esquerdo do mouse na barra lateral esquerda, exatamente na linha onde se deseja que o programa pare

a execução. Ao clicar, um ponto vermelho será adicionado e a linha ficará destacada em vermelho, conforme figura a seguir.



Com o *break point* posicionado, e apertando F5, o programa será executado até a linha anterior a linha em vermelho, conforme figura a seguir.



Assim, o código pode continuar a ser executado linha a linha por meio da tecla F8 ou na sua totalidade por meio da tecla F5. *Break points* são úteis para se executar partes do código de uma vez só (em especial onde sabe-se que não contém erros). Diversos *Break Points* podem ser colocados em um programa.

8. VETORES

Um vetor (ou matriz unidimensional) é uma sequência linear de elementos armazenados consecutivamente na memória do computador.

São variáveis do mesmo tipo, declaradas com o mesmo nome e referenciadas por um índice (também chamado de posição, que é sempre um número inteiro) para determinar sua localização dentro da estrutura.

Por padrão o índice no VBA começa em 0. Porém, índices começando em 1 são mais intuitivos. Para um índice começar em 1 no VBA, usa-se (antes das subrotinas e funções) o seguinte comando:

```
Option Base 1
```

Declaração de vetores:

```
Dim v(tamanho) As Tipo
```

Onde `tamanho` é o número de elementos do vetor `v`, e `Tipo` é o tipo de dado que o vetor irá conter.

Por exemplo,

```
Dim v(10) As Integer
```

cria um vetor de números inteiros chamado `v` de tamanho 10 (10 posições).

E para atribuir valores aos 10 elementos de `v` basta acessar o índice do vetor por meio dos parênteses:

```
v(1) = 5
v(2) = 45
v(7) = -4
v(10) = 300
```

Neste exemplo, apenas as posições 1, 2, 7 e 10 foram modificadas, as demais continuam contendo o valor inicial (que é zero).

Este exemplo de vetor é chamado de **estático**, pois uma vez declarado com um tamanho fixo não é mais possível mudar o seu tamanho.

Porém, se não se sabe antecipadamente o tamanho de um vetor, ou se o vetor poderá mudar de tamanho ao longo do programa, então usa-se um vetor **dinâmico**, declarado da seguinte forma:

```
Dim v() As Tipo
```

E após descoberto o tamanho do vetor (por exemplo, um certo valor de n), deve-se redimensionar este vetor da forma:

```
ReDim v(n)
```

Importante: um vetor dinâmico pode receber valores somente depois de redimensionado.

O redimensionamento de um vetor pode acontecer em qualquer momento no programa: quando o vetor ainda está vazio ou quando ele já tem um tamanho e está preenchido.
Entretanto, no caso em que um vetor já tem um tamanho e já contém valores, e acontece um redimensionamento usando `ReDim`, os valores são perdidos e zeros são colocados em seus lugares.
Para preservar os valores originais, usa-se o comando `ReDim Preserve v(n)` ao invés de apenas `ReDim v(n)`.

O primeiro e o último índice de um vetor podem ser obtidos, respectivamente, por meio dos comandos:

```
LBound(v)
UBound(v)
```

Para o exemplo a seguir, com `Option Base 1`:

```
Dim v(34) As Boolean
```

Tem-se que `LBound(v)` vale 1 (primeiro índice) e `UBound(v)` vale 34 (último índice).

Geralmente o `LBound` de um vetor vale 1, exceto os casos em que a declaração é feita, por exemplo, da forma:

```
Dim v(4 To 12) As Single
```

Neste caso, o `LBound(v)` vale 4 e `UBound(v)` vale 12. Este vetor não tem as três primeiras posições e o tamanho dele é 9 (`UBound-LBound+1`). Porém esta forma de definir vetores é incomum.

Exemplo: Criar uma função que tenha dois vetores como parâmetro. Retornar o produto internos desses dois vetores, quando possível.

```
Function ProdutoInterno(v1 As Single, v2 As Single) As Variant
     Dim i As Integer
     If UBound(v1) <> UBound(v2) Then
          ProdutoInterno = "Impossível calcular o produto interno"
     Else
          For i = 1 To UBound(v1)
               ProdutoInterno = ProdutoInterno + v1(i) * v2(i)
          Next
     End If
End Function
```

Esta função tem como retorno uma variável `Variant`, pois em uma situação ela pode ser uma string e em outra ela pode ser um valore real.

Quando um vetor é passado como parâmetro para uma subrotina ou função, se esse vetor sofrer modificações dentro desta subrotina ou função, essas modificações serão permanentes, ou seja, após o final da execução da subrotina ou função, o vetor será retornado à subrotina original com as alterações que ele veio a sofrer. Isso acontece porque para vetores (e matrizes vistas adiante) não é possível usar o modo `ByVal` para passagem de parâmetros.

Exemplo: Criar uma subrotina que, dado um vetor `v` como parâmetro, retorne dois outros vetores: um com os elementos negativos e nulos de `v` e outro com os elementos positivos de `v`.

```
Sub Negativos_Positivos(v As Single, vn As Single, vp As Single)
     Dim nn As Integer, np As Integer, i As Integer
```

```
    'descobrindo o número de elementos negativos e positivos de v
    For i = 1 To UBound(v)
        If v(i) <= 0 Then
            nn = nn +1
        Else
            np = np +1
        End If
    Next
    'redimensionando os dois vetores para os tamanhos corretos
    ReDim vn(nn)
    ReDim vp(np)

    'zerando os valores de nn e np, pois serão utilizados novamente
    nn = 0
    np = 0

    'atribuindo valores aos vetores
    For i = 1 To UBound(v)
        If v(i) <= 0 Then
            nn = nn +1
            vn(nn) = v(i)
        Else
            np = np +1
            vp(np) = v(i)
        End If
    Next
End Sub
```

Neste exemplo, o vetor `v` tem tamanho e está preenchido com valores. Já os vetores `vn` e `vp` entram vazios e sem tamanho na subrotina, e dentro dela recebem o redimensionamento e elementos. Assim, uma outra subrotina criada para chamar `Negativos_Positivos` deve ter criado `vn` e `vp` e ao terminar a execução terá `vn` e `vp` preenchidos e com tamanho.
Obs: Neste exemplo, poderia ter se usado `ReDim Preserve`, o que tornaria o código bem mais enxuto, porém preservar valores é computacionalmente mais caro, por este motivo sempre prefira usar `ReDim` ao invés de `ReDim Preserve`, quando possível.

9. MATRIZES

Matrizes são extensões de vetores, onde se tem mais que uma dimensão. Matrizes podem ter muitas dimensões, porém aqui será estudado o caso particular de duas dimensões.

A declaração em VBA para matrizes bidimensionais é feita da forma:

```
Dim M(linhas,colunas) As Tipo
```

Onde `linhas` e `colunas` são o número de linhas e de colunas, respectivamente, que a matriz `M` terá. `Tipo` é o tipo de dado da matriz.

Por exemplo:

```
Dim M(2,3) As Integer
```

Aqui está sendo criada uma matriz chamada `M`, de números inteiros, que tem 2 linhas e 3 colunas. A matriz `M` foi criada como sendo fixa, ou seja, não se pode alterar suas dimensões.

Para atribuir valores a esta matriz basta usar a indexação dos índices das respectivas linhas e colunas. Por exemplo:

```
M(1,1) = 1
M(1,2) = -8
M(1,3) = 5
M(2,1) = 0
M(2,2) = 9
M(2,3) = -2
```

Os comandos `LBound` e `UBound` também podem ser usados para matrizes. Porém, além do parâmetro nome, deve ser informado um segundo parâmetro, que deve ser 1 para linha ou 2 para coluna.
`UBound(M,1)` fornece o número de linhas que a matriz `M` tem e `UBound(M,2)` o número de colunas da matriz `M`.
Para o exemplo anterior, `UBound(M,1)` vale 2 e `UBound(M,2)` vale 3.

Matrizes fixas são usadas quando já se sabe suas dimensões e não se pode alterá-las. Porém matrizes dinâmicas são muito mais utilizadas, e no VBA são declaradas da forma:

```
Dim M() As Tipo
```

Ou seja, uma matriz dinâmica é declarada igual a um vetor dinâmico, e antes de receber valores, ela deve ser redimensionada, por meio do comando `ReDim`. Porém, neste caso, o comando `ReDim`, deve receber dois parâmetros: o número de linhas e o número de colunas da matriz.

```
ReDim M(10,6)
```

Neste exemplo, a matriz `M` conterá 10 linhas e 6 colunas.
Assim como em vetor, matriz dinâmica só pode receber elementos se ela já foi redimensionada.
Uma matriz dinâmica pode ser redimensionada em qualquer momento, porém o comando `ReDim` zera todos os elementos. O `ReDim Preserve` para matrizes só pode ser usado para alterar a última dimensão (não é comumente usado).

Exemplo: Criar e imprimir no Excel uma matriz de números aleatórios entre 0 e 100, onde o usuário informa as dimensões da matriz.

```
Sub Cria_matriz_aleatoria()
     Dim Matriz() As Single
     Dim lin As Integer, col As Integer
     Dim i As Integer, j As Integer

     lin = InputBox("Insira o número de linhas da matriz")
     col = InputBox("Insira o número de colunas da matriz")

     Redim Matriz(lin,col)

     For i = 1 To lin
          For j = 1 To col
               Matriz(i,j) = Rnd * 100
               Cells(i,j) = Matriz(i,j)
          Next j
     Next i
End Sub
```

Obs: Como neste exemplo as dimensões serão informadas pelo usuário (variáveis), não é possível ler estes valores e depois fazer `Dim Matriz(lin,col) As Single`. Primeiro é preciso dimensionar a matriz dinâmica e depois redimensioná-la.

Assim como acontece com vetores, quando uma matriz é passada como parâmetro para uma subrotina ou função, se essa matriz sofrer modificações dentro desta subrotina ou função, essas modificações serão permanentes, ou seja, após o final da execução da subrotina ou função, a matriz será retornada à subrotina original com as alterações que ela veio a sofrer. Isso acontece porque para matrizes não é possível usar o modo `ByVal` para passagem de parâmetros.

10. LEITURA E GRAVAÇÃO DE ARQUIVOS DE TEXTO

10.1 Leitura

Para abrir, ler e fechar um arquivo de texto em VBA, usa-se:

```
Open "nome_arquivo" For Input As #id
Input #id, nome_variavel
Close #id
```

Onde:

- `nome_arquivo:` expressão string que especifica o nome do arquivo, juntamente com o seu diretório;
- `id:` número entre 1 e 511 que é utilizado como identificador;
- `nome_variavel:` nome da variável que vai receber o dado do arquivo.

Na primeira linha, o comando `Open` abre o arquivo, o comando `For Input` indica que o arquivo foi aberto para leitura. Na segunda linha, o comando `Input` faz a leitura de um dado do arquivo (um único dado pode ser lido por vez). Na terceira linha, o comando `Close` fecha o arquivo.

Por exemplo, um arquivo chamado `Teste.txt`, que está no diretório C:\Users\UFPR\Programacao, contém dois números (eles podem estar separados por um espaço, separados por vírgula ou em linhas diferentes). Para ler estes dois valores, basta fazer:

```
Sub LerArquivo()
     Dim a As Single, b As Single
     Open "C:\Users\UFPR\Programacao\Teste.txt" For Input As #15
     Input #15, a
     Input #15, b
     Close #15
End Sub
```

Neste exemplo, o `id` utilizado foi o 15 (escolhido aleatoriamente). As variáveis `a` e `b` receberam, respectivamente, os dois primeiros valores do arquivo (alocados em variáveis do tipo `Single`). Pode haver mais valores no arquivo, porém somente dois foram lidos.

Para ler um número real, esse número deve estar com ponto como separador das casas decimais no arquivo de texto.

Dois arquivos podem ser abertos simultaneamente, entretanto os `id's` devem ser diferentes:

```
Sub Ler2Arquivos()
     Dim a As Single, b As Single
     Open "C:\Users\UFPR\Programacao\Teste1.txt" For Input As #3
     Open "C:\Users\UFPR\Programacao\Teste2.txt" For Input As #8
     Input #3, a
     Input #8, b
     Close #3
     Close #8
End Sub
```

Neste exemplo, dois arquivos diferentes foram abertos, com `id's` diferentes (3 e 8). A variável `a` recebeu o primeiro valor do arquivo `Teste1.txt` e a variável `b` recebeu o primeiro valor do arquivo `Teste2.txt` (é possível saber pelo número do `id` utilizado ao lado do comando `Input`).

10.2 Comando EOF(id)

Quando se quer ler um arquivo mas não se sabe qual é a quantidade de dados que há nele, pode-se usar o comando `EOF(id)` (E=end, O=of, F=file). É um comando booleano que dá como resposta `True` se o final do arquivo foi alcançado e `False` caso contrário.

Obs: A tecla ENTER conta como zero para números, " " para strings e `False` para booleanos.
Um ponto e vírgula faz com que a leitura de dados daquela linha seja interrompida e a leitura continua na linha seguinte.

Exemplo: Pode-se utilizar o comando `EOF` para ler os dados do arquivo com uma estrutura de repetição e colocá-los em um vetor.

```
Sub texto()
     Dim v() As Single
     Dim i As Integer
     Dim aux As Single


     Open "C:\Users\UFPR\Programacao\Teste.txt" For Input As #10

     While EOF(10) = False
         i = i + 1
         Input #10, aux
     Wend
     Close #10

     ReDim v(i)

     Open "C:\Users\UFPR\Programacao\Teste.txt" For Input As #10
     For i = 1 To UBound(v)
          Input #10, v(i)
     Next
     Close #10
End Sub
```

No Código anterior, a variável `aux` recebe um dado de cada vez do arquivo, mas não tem utilidade alguma, pois a cada iteração seu valor é sobrescrito. Ele serve apenas para receber o dado e a leitura pular para a próxima linha, pois o que está sendo encontrado é o número de dados que o arquivo contém (a variável `i` está fazendo essa contagem no `While`).
Após descoberto o número de elementos no arquivo, o vetor é redimensionado e então recebe valores. Note que o arquivo precisou ser aberto duas vezes neste código (foi usado o mesmo `id`, mas não é regra), pois no primeiro laço (`While`) se chegou no final do arquivo e para se atribuir os valores ao vetor (com o laço do `For`), foi necessário reabri-lo.

Se fosse utilizado o `ReDim Preserve` (mais custoso computacionalmente) ao invés do `ReDim`, o código ficaria assim:

```
Sub texto_ReDimPreserve()
     Dim v() As Single
     Dim i As Integer

     Open "C:\Users\UFPR\Programacao\Teste.txt" For Input As #10

     While EOF(10) = False
         i = i + 1
         ReDim Preserve v(i)
         Input #10, v(i)
     Wend

     Close #10
End Sub
```

O vetor resultante será composto por todos os dados presentes no arquivo `Teste.txt` e será redimensionado ao longo da estrutura de repetição em uma posição a mais até que se atinja o final do arquivo.

10.3 Gravação

Para abrir, escrever e fechar um arquivo de texto em VBA, usa-se:

```
Open "nome_arquivo" For Output As #id
Print #id, nome_variavel
Close #id
```

Onde `nome_arquivo`, `id` e `nome_variavel` são os mesmos já definidos.

Na primeira linha, o comando `Open` abre o arquivo, o comando `For Output` indica que o arquivo foi aberto para gravação (escrita). Na segunda linha, o comando `Print` faz a impressão de um dado do arquivo (um único dado pode ser impresso por vez). Na terceira linha, o comando `Close` fecha o arquivo.

Esta forma de gravação imprime cada dado em uma linha do arquivo. Para escrever dados em uma mesma linha, usa-se ponto e vírgula depois do `nome_variavel`.

Se um arquivo que foi aberto para impressão não existir ainda, ele será criado. Se ele já existe, o arquivo será substituído.

Para imprimir dados em um arquivo que já contenha dados, preservando os dados existentes, usa-se o comando `Append` ao invés do comando `Output`.

```
Open "nome_arquivo" For Append As #id
```

11. MANIPULAÇÃO DE STRINGS

As principais funções para manipulação de strings no VBA são:

FUNÇÃO	SIGNIFICADO
`&`	Concatena números e strings
`Len(x)`	Número de caracteres de `x`
`Mid(x,n)`	Extrai de `x` a `n`-ésima componente em diante
`Mid(x,n,c)`	Extrai de `x` a `n`-ésima componente em diante, porém de comprimento `c`
`Left(x,n)`	Extrai de `x` as `n` primeiras componentes
`Right(x,n)`	Extrai de `x` as `n` últimas componentes
`InStr(x,y)`	Primeiro índice onde aparece a string `y` em `x`
`Replace(x, y, z)`	Substitui em `x` todos as ocorrências da string `y` pela string `z`
`LTrim(x)`	Retira espaços em branco de `x` à esquerda
`RTrim(x)`	Retira espaços em branco de `x` à direita
`LCase(x)`	Transforma todos os caracteres de `x` em letra minúscula
`UCase(x)`	Transforma todos os caracteres de `x` em letra maiúscula
`Asc(y)`	Retorna um número inteiro correspondente a string `y` na tabela ASCII
`Chr(n)`	Retorna o caractere correspondente ao número `n` na tabela ASCII
`Val("n ")`	Transforma "`n` " em valor numérico
`CStr(n)`	Transforma o valor numérico `n` em string
`Split(x,y)`	Separa `x` pela string `y` (o resultado é um vetor com índice começando no 0)
`Join(v,y)`	Junta o conteúdo de um vetor `v`, agrupando os elementos, colocando como agregador a string `y` (o resultado é uma string)

Obs1: `x` é uma variável do tipo String, `n` e `c` são valores inteiros, `y` e `z` são quaisquer strings.

Obs2: Para algumas funções descritas anteriormente, é mais comum `x` ser uma variável string, mas funciona também se `x` é um número. Por exemplo, `Left(231546, 2)` retorna o número 23. Em resumo, o parâmetro `x` foi dimensionado como `Variant` no momento da criação de algumas dessas funções.

Obs 3: A tabela ASCII é uma tabela que representa letras e caracteres de pontuação por meio de um código binário de 8 bits. A tabela pode ser encontrada no link https://www.ascii-code.com

Exemplo: Suponha que no Excel, nomes completos de clientes de uma empresa estão digitados na primeira coluna da planilha. Descobrir quantos clientes tem *Silva* em seu sobrenome (pode ser no meio ou no final do nome).

```
Sub quantidade_sobrenome_silva()
    Dim i As Integer, qtd As Integer
    i = 1
    While Cells(i, 1) <> ""
        If InStr(LCase(Cells(i, 1)), " silva ") > 0 Or _
        Right(LCase(Cells(i, 1)), 5) = "silva" Then
            qtd = qtd + 1
        End If
        i = i + 1
    Wend
    MsgBox ("A quantidade de clientes que tem Silva no sobrenome é " _
    & qtd)
End Sub
```

Neste exemplo, o conteúdo das células foi transformado para fonte em letra minúscula por meio da função `LCase`, pois a palavra *Silva* pode estar digitada de várias formas (*Silva*, *SILVA*, *silva*, *SILva*,..). A primeira condição colocada no `If` é se a função `InStr` retornar um valor maior que zero, significa que encontrou a palavra no meio do nome da pessoa (observe os espaços antes e depois da palavra, e neste caso ele não irá encontrar por exemplo a palavra *silvana*). A segunda condição do `If` é se os cinco últimos caracteres de toda a célula são iguais a *silva* (neste caso *Silva* é o último nome da pessoa). Veja o operador lógico `Or` no `If`, pois qualquer uma das duas condições pode acontecer.

Exemplo: Ainda usando os mesmos dados do exemplo anterior, agora imprima na segunda coluna do Excel somente o primeiro nome de cada cliente.

```
Sub Primeiro_nome()
    Dim i As Integer, aux As String, n As Integer
    i = 1
    While Cells(i, 1) <> ""
        aux = Cells(i, 1)
        n = InStr(aux, " ")
        Cells(i, 4) = Left(aux, n - 1)
        'Cells(i, 4) = Mid(aux, 1, n - 1)
        i = i + 1
    Wend
End Sub
```

Neste exemplo, a função `InStr` retorna o número do caractere onde o primeiro espaço em branco aparece na célula, ou seja, o fim no primeiro nome. A função `Left` retorna os *n-1* primeiros caracteres

da célula, pois *n* é o valor correspondente ao caractere em branco. A função `Mid` também poderia ser usada neste exemplo.

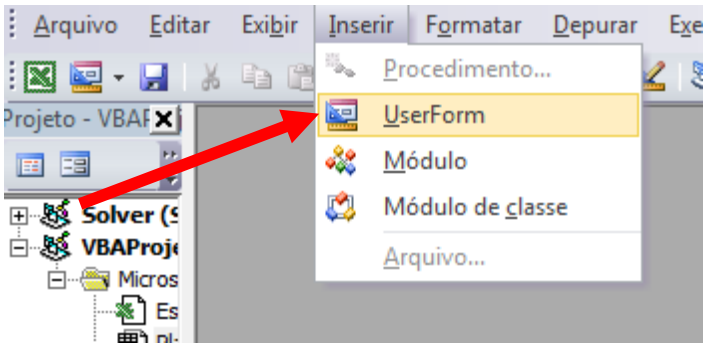
12. FORMULÁRIOS NO VBA

Os formulários (UseForms) combinam as capacidades de `InputBox` e `MsgBox` para criar uma maneira mais eficiente de interagir com o usuário. Para isso o formulário pode ser construído para receber informações, executá-las e mostrar os resultados ao usuário.

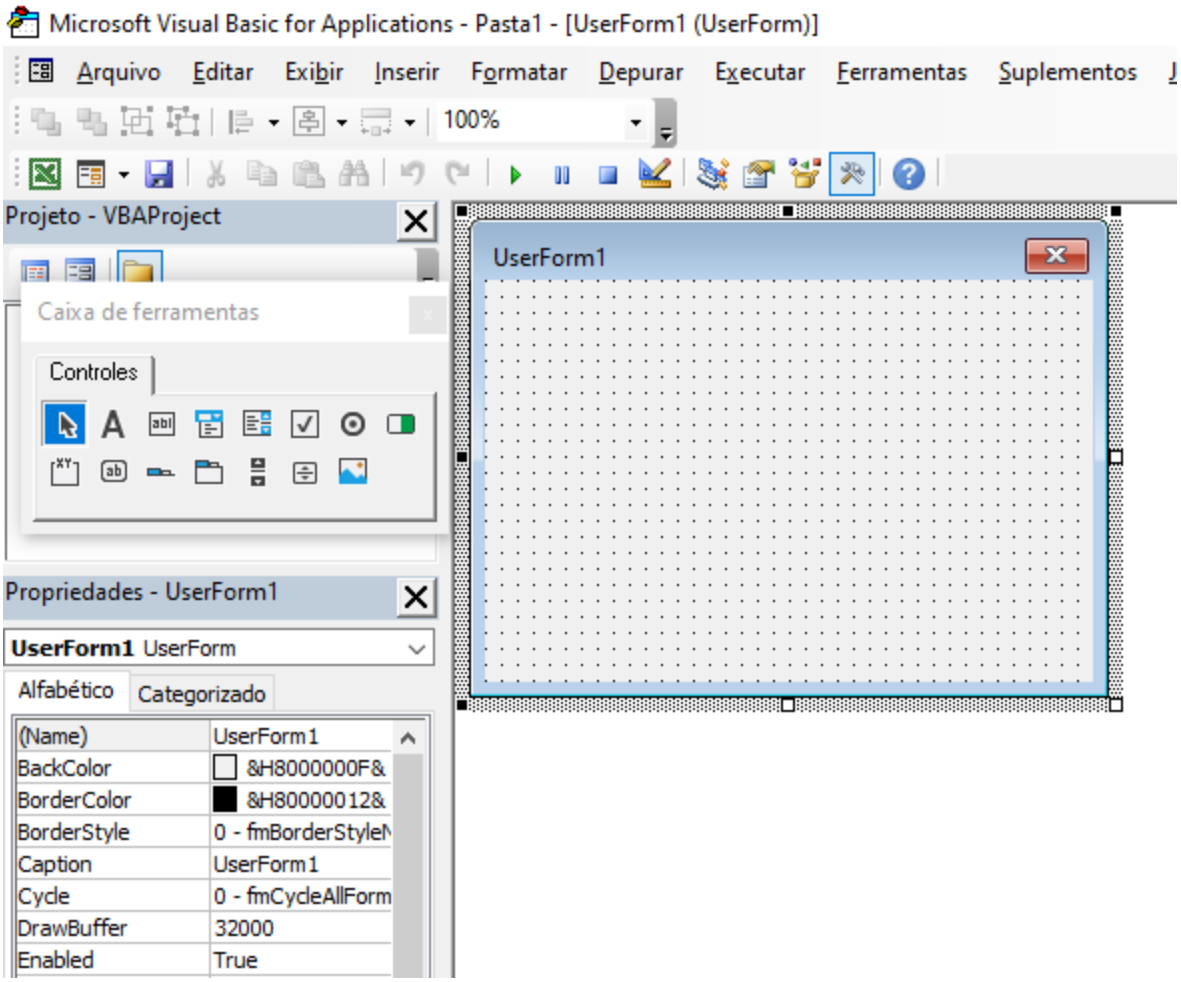
Exemplo: Criar um formulário que calcula o índice de massa corporal de uma pessoa, onde são fornecidos o nome (apenas por curiosidade, pois esse dado não é usado), peso (Kg) e altura (m), e como resultado é apresentado o resultado do IMC no formulário.

No editor do VBA:

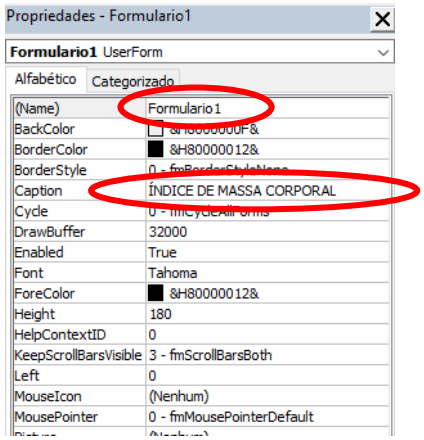
- Inserir → UserForm



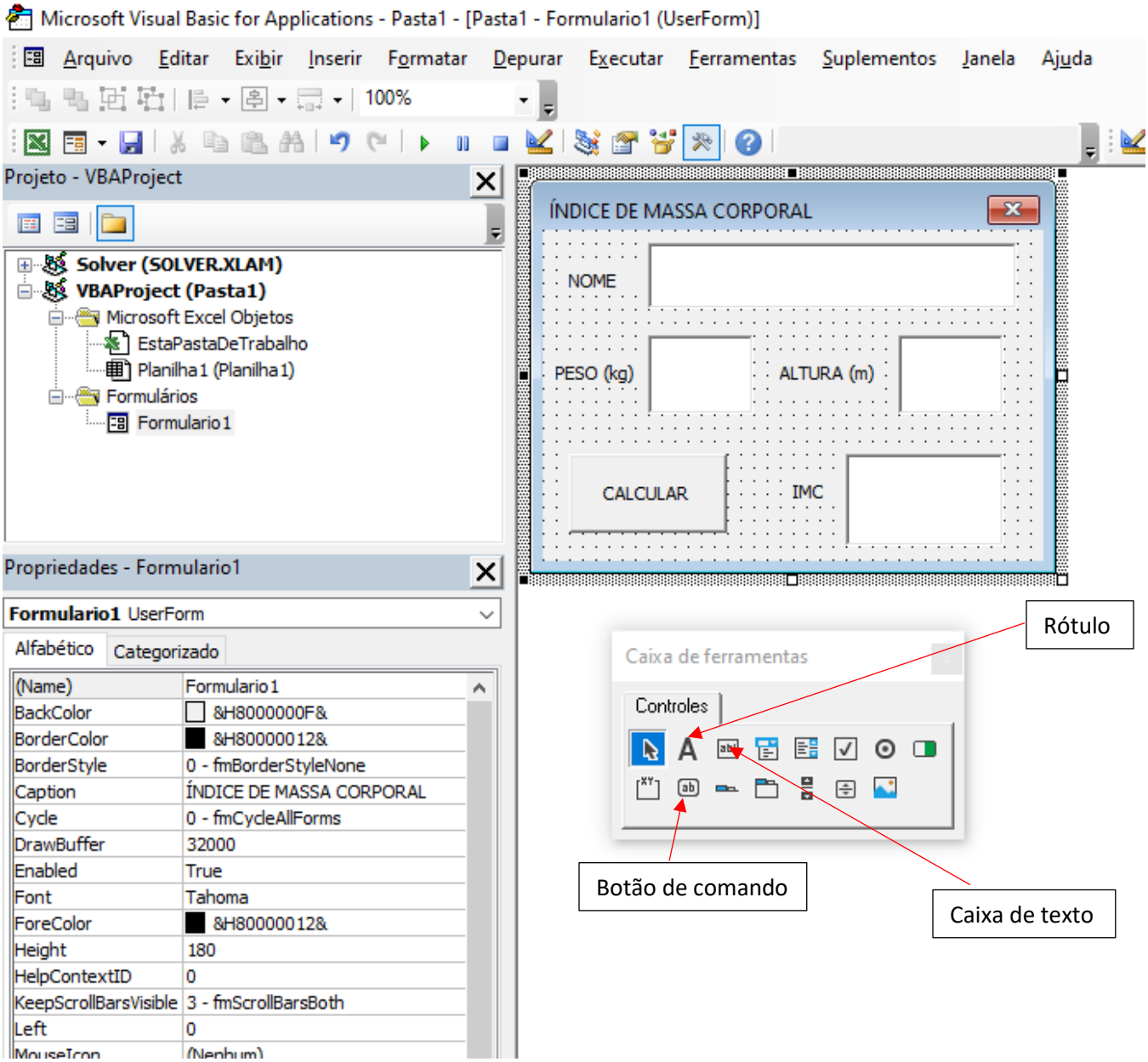
O resultado será a seguinte tela:



Na janela "Propriedades" é possível trocar, por exemplo, o nome (*Name*) e o título (*Caption*) do formulário. Neste exemplo, será usado o nome *Formulario1* e o título *ÍNDICE DE MASSA CORPORAL.*



Por meio da janela “Caixa de ferramentas” (se ela não estiver visível, vá em Exibir → Caixa de ferramentas), é possível inserir caixa de texto, rótulo e botão de comando.



Dar duplo clique para inserir código nos botões de comando. O seguinte programa será exibido:

```
Private Sub CommandButton1_Click()


End Sub
```

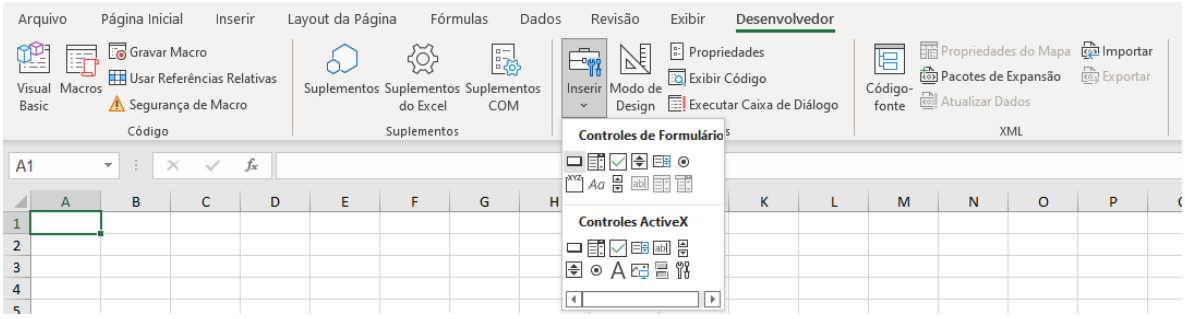
Então, adicionam-se as ações desejadas ao botão.

```
Private Sub CommandButton1_Click()
   Dim IMC As Single
   Dim nome As String
   Dim peso As Single, altura As Single

   'Atribuindo valores às variáveis
   nome = TextBox1.Text
   peso = TextBox2.Text
   altura = TextBox3.Text

   'Cálculo do IMC
   IMC = peso / altura ^ 2

   ' Apresentando o resultado no formulário
   TextBox4 = IMC
End Sub
```

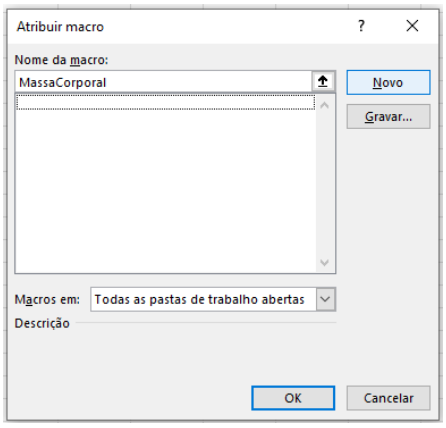
No programa anterior, as caixas de texto por padrão recebem o nome `TextBox` seguido de um número, que é a ordem com que foram inseridos no formulário. Este nome pode ser alterado na aba "Propriedades" de cada objeto.

Então cria-se um botão no Excel para ativar o formulário:

Desenvolvedor → Inserir → Controles de Formulário → Botão.



Desenhar o botão e atribuir a macro de ativação.



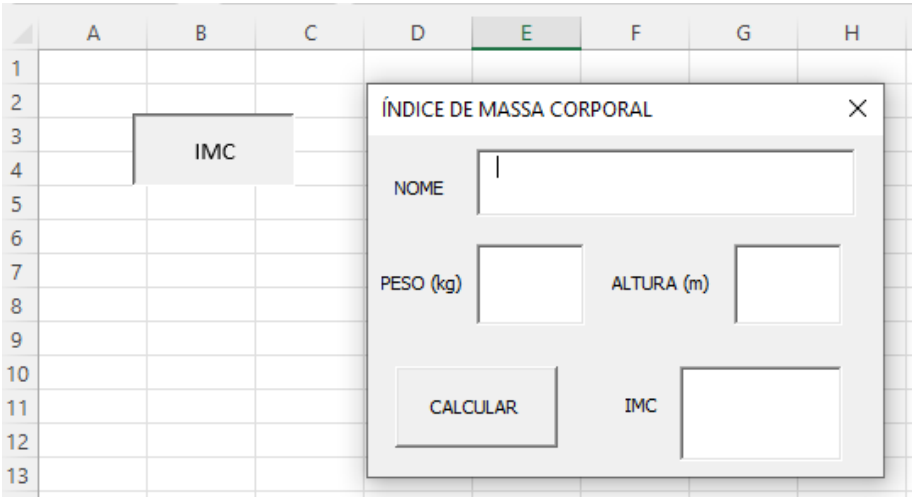
Na figura anterior, o nome da subrotina será `MassaCorporal`. Clicar em Novo, onde abrirá a seguinte subrotina (dentro de Módulos):

```
Sub MassaCorporal()

End Sub
```

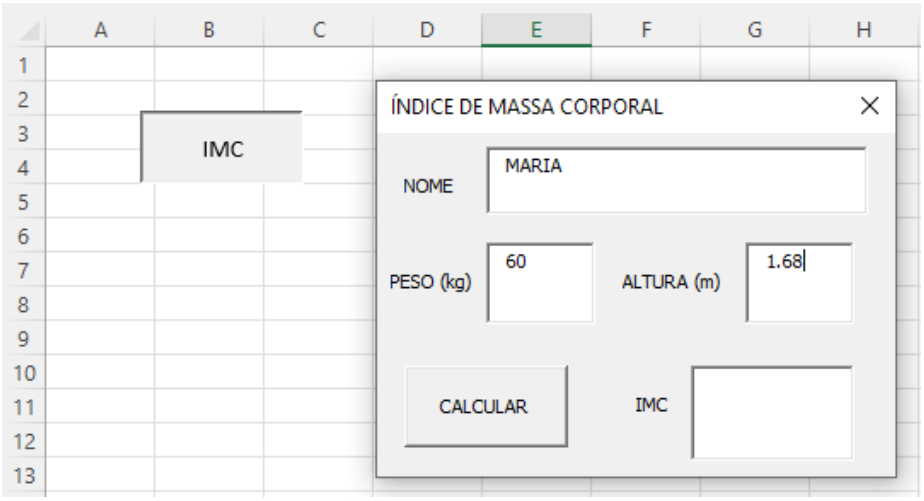
Então é acrescentado à subrotina o comando que abre o formulário quando o botão é apertado:

```
Sub MassaCorporal()
     Formulario1.Show
End Sub
```

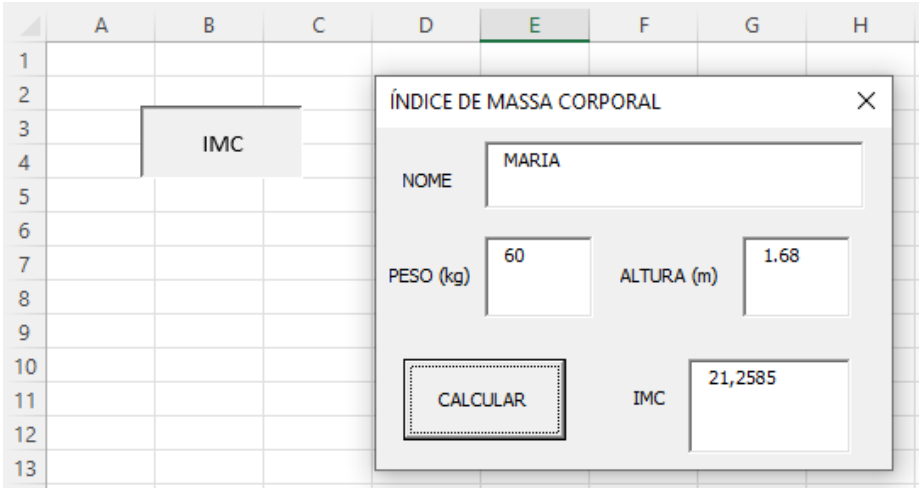
No Excel, o texto do botão pode ser alterado. Neste exemplo, o botão no Excel recebeu o nome de IMC. Ao apertar o botão, aparecerá a seguinte tela:



Aqui, nome, peso e altura podem ser inseridos. Verificar se na versão do Excel que se está usando, a inserção de números reais é com ponto ou vírgula (assim como no `InputBox`).



Após a inserção dos dados, aperta-se o botão CALCULAR, e o resultado é impresso.



13. LISTA DE EXERCÍCIOS

Estes exercícios foram retirados da internet, sendo feito um compilado de diversos sites.
Importante destacar que não existe um "gabarito" para exercícios de programação, pois podem ser feitos de diversas maneiras.

14.1 Exercícios iniciais

1. Faça uma subrotina que mostre a mensagem "Alô mundo" na tela.

2. Faça uma subrotina que peça um número e então mostre a mensagem: O número informado foi [número].

3. Faça uma subrotina que peça dois números e imprima a soma.

4. Escreva uma subrotina que calcule a média aritmética entre dois números.

5. Faça uma subrotina que converta metros para centímetros.

6. Faça uma subrotina que dado o tamanho do lado de um quadrado, calcular a área e o perímetro deste quadrado.

7. Faça uma subrotina que peça o raio de um círculo, calcule e mostre sua área.

8. Dados os comprimentos dos catetos de um triângulo retângulo, criar uma subrotina para determinar e imprimir o comprimento da hipotenusa.

9. Faça um programa que peça a temperatura em graus Fahrenheit, transforme e mostre a temperatura em graus Celsius. A transformação é feita por: $C = 5 * (\frac{F-32}{9})$.

10. Faça um programa que peça a temperatura em graus Celsius, transforme e mostre em graus Fahrenheit.

11. Faça um programa que pergunte quanto você ganha por hora e o número de horas trabalhadas no mês. Do seu salário bruto são descontados 11% para o Imposto de Renda, 8% para o INSS e 5% para o sindicato. Apresente os resultados de:

- Salário bruto;
- Quanto pagou ao Imposto de Renda;
- Quanto pagou ao INSS;
- Quanto pagou ao sindicato;
- Salário líquido (salário líquido = salário bruto - descontos).

12. Escreva um programa para ler as dimensões de uma cozinha retangular (comprimento, largura e altura), calcular e escrever a quantidade de caixas de azulejos para se colocar em todas as suas paredes (considere que não será descontada a área ocupada por portas e janelas). Cada caixa de azulejos tem 1,5 m^2 .

13. Faça um programa que simula o lançamento de um dado. O valor sorteado deve ser impresso na tela. Dica: use a função *Rnd* do VBA .

14. João Papo-de-Pescador, homem de bem, comprou um microcomputador para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do estado de São Paulo (50 quilos) deve pagar uma multa de R$ 4,00 por quilo excedente. João precisa que você faça um programa que leia a variável peso (peso de peixes) e calcule o excesso. Gravar na variável excesso a quantidade de quilos além do limite e na variável multa o valor da multa que João deverá pagar. Imprima os dados do programa com as mensagens adequadas.

15. Escreva um programa no qual é informado o valor do saque realizado pelo cliente de um banco. Imprima quantas notas de cada valor serão necessárias para atender ao saque com a menor quantidade de notas possível. Serão utilizadas notas de 100, 50, 20, 10, 5, 2 e 1 real.

16. Faça um programa que leia um número inteiro menor que 1000 e imprima a quantidade de centenas, dezenas e unidades desse número.

- Observando os termos no plural, colocação do "e" e da vírgula.
- Exemplos: 326 = 3 centenas, 2 dezenas e 6 unidades; 12 = 1 dezena e 2 unidades.
- Testar com: 326, 300, 100, 320, 310,305, 301, 101, 311, 111, 25, 20, 10, 21, 11, 1, 7 e 16.

17. Informe o tipo e o valor de cada variável.

```
A1 = 3 / 2
a2 = 3 Mod 2
a3 = 3 \ 2
a4 = 20 – 12 Mod 5
a5 = 5 / 4 * 2
a6 = a1 <> 1.5
a7 = 5 = 6
a8 = 5 = 6 Or True
a9 = 5 > 10 And a2 = a2
a10 = Not False
a11 = “oi”
a12 = “Eu tenho “ & a4 & “ anos”
a13 = 2 * 0.7 * 10^308
```

14.2 Exercícios de estruturas de decisão (condicionais)

18. Escrever uma subrotina que imprima uma mensagem na tela informando o maior número: se o que está na célula D9, o que está na célula F6 ou se são iguais.

19. As maçãs custam R$ 1,30 cada se forem compradas menos de uma dúzia, e R$ 1,00 se forem compradas pelo menos 12. Escreva um programa que leia o número de maçãs compradas, calcule e escreva o custo total da compra.

20. Faça uma subrotina que peça um valor e mostre na tela se o valor é positivo, negativo ou nulo.

21. Dados três números naturais, verificar se eles formam os lados de um triângulo retângulo.

22. Faça um programa que peça os 3 lados de um triângulo. O programa deverá informar se os valores podem ser um triângulo. Indique, caso os lados formem um triângulo, se ele é: equilátero, isósceles ou escaleno.
Dicas:

- Três lados formam um triângulo quando a soma de quaisquer dois lados é maior que o terceiro;
- Triângulo Equilátero: três lados iguais;
- Triângulo Isósceles: quaisquer dois lados iguais;
- Triângulo Escaleno: três lados diferentes.

23. Faça um programa que leia um número e exiba o dia correspondente da semana (1-Domingo, 2-Segunda, etc.). Se digitar outro valor deve aparecer “Valor inválido”.

24. Escreva uma função que dado um parâmetro do tipo inteiro, devolve *True* se o argumento for um inteiro par e *False* se for um inteiro ímpar.

25. Faça um programa que verifique se uma letra digitada é vogal ou consoante.

26. Tendo como dados de entrada a altura (em metros) e o sexo de uma pessoa, construa um algoritmo que calcule seu peso ideal, utilizando as seguintes fórmulas:

- Para homens: (72.7*h) - 58
- Para mulheres: (62.1*h) - 44.7

O resultado do algoritmo deve ser uma mensagem informando se a pessoa está acima, abaixo ou no peso ideal.

27. Dados os coeficientes *a, b* e *c* de uma equação de segundo grau, verificar se existe uma raiz real, duas raízes reais ou nenhuma raiz real. Os coeficientes devem ser lidos com a função *InputBox*, e o resultado deve ser da forma: na célula A1 deve constar “Uma raiz real” e em A2 o valor desta raiz, ou “Duas raízes reias” em A1 e os seus valores em A2 e A3 ou “Nenhuma raiz real” em A1.

28. Criar uma subrotina que calcula a nota de um aluno em certa disciplina onde a forma de avaliação se dá por meio de três provas. A primeira prova tem peso 3, a segunda prova tem peso 4 e a terceira tem peso 5. Essas notas encontram-se nas células A1, B1 e C1 respectivamente. Além da nota final, que deve ser apresentada na célula A3, também deve ser informada (na célula B3) a situação do aluno, onde *Aprovado* significa média maior ou igual a 70,00, *Exame* significa média inferior a 70,00 e maior ou igual a 40,00 e *Reprovado* significa média inferior a 40,00.

29. Seja o seguinte algoritmo escrito em pseudo-código (não escrito em uma linguagem de programação específica):

```
início
      ler x
      ler y
      z ← (x*y) + 5
      se z <= 0 então
        resposta ← 'A'
      senão
        se z <= 100 então
              resposta ← 'B'
        senão
              resposta ← 'C'
        fim_se
      fim_se
      escrever  z, resposta
fim
```

Execute o algoritmo no papel e complete o quadro a seguir para os seguintes valores:

Variáveis			
X	**Y**	**Z**	**Resposta**
3	2		
150	3		
7	-1		
-2	5		
50	3		

30. Faça um programa que leia três números inteiros positivos e efetue o cálculo de uma das seguintes médias de acordo com um valor inteiro digitado pelo usuário:

- Se o número 1 é digitado então a média geométrica deve ser calculada: $\sqrt[3]{x * y * z}$
- Se o número 2 é digitado então a média ponderada deve ser calculada: $\frac{x+2*y+3*z}{6}$
- Se o número 3 é digitado então a média harmônica deve ser calculada: $\frac{1}{\frac{1}{x}+\frac{1}{y}+\frac{1}{z}}$

- Se o número 4 é digitado então a média aritmética deve ser calculada: $\frac{x+y+z}{3}$

31. Uma empresa quer verificar se um empregado está qualificado para a aposentadoria ou não. Para estar em condições, pelo menos um dos seguintes requisitos deve ser satisfeito:

- Ter no mínimo 65 anos de idade;
- Ter trabalhado no mínimo 30 anos;
- Ter no mínimo 60 anos e ter trabalhado no mínimo 25 anos.

Com base nestas informações, faça um algoritmo que leia o ano do nascimento de um empregado e o ano de seu ingresso na empresa. O programa deverá escrever a idade e o tempo de trabalho do empregado e a mensagem "Requer aposentadoria" ou "Não requer aposentadoria".

32. Um posto de gasolina está vendendo combustíveis com a seguinte tabela de descontos:

ÁLCOOL	Até 20 litros, desconto de 3% por litro
	Acima de 20 litros, desconto de 5% por litro
GASOLINA	Até 20 litros, desconto de 4% por litro
	Acima de 20 litros, desconto de 6% por litro

Escreva um algoritmo que leia o número de litros vendidos e o tipo de combustível (A-Álcool ou G-Gasolina), calcule e imprima o valor a ser pago pelo cliente sabendo que o preço do litro da gasolina é R$5,67 e o preço do litro do álcool é R$4,67.

33. Faça um algoritmo que leia dois números reais. Após isso, apresente o seguinte menu e execute a operação escolhida. Escreva uma mensagem de erro se a opção for inválida.
Escolha a opção:
1 - Soma de dois números
2 - Diferença entre dois números (maior pelo menor)
3 - Produto entre dois números
4 - Divisão entre dois números (o denominador não pode ser zero)
Digite a opção:

34. Escreva uma subrotina que dados 4 valores, retorne a média aritmética desses valores descartando o menor valor e o maior valor.

35. Faça um programa que leia três números e mostre-os em ordem decrescente.

36. Leia uma data no formato DD/MM/AAAA e verifique se é uma data válida.

37. Faça um algoritmo que, dados três números inteiros representando dia, mês e ano de uma data, imprima qual é o dia seguinte:

- **São bissextos** todos os anos múltiplos de 400, por ex.: **1600, 2000, 2400, 2800...**

- **São bissextos** todos os múltiplos de 4, exceto se for múltiplo de 100 mas não de 400, por ex.: **1996, 2000, 2004, 2008, 2012, 2016, 2020, 2024, 2028...**

14.3 Exercícios de estruturas de repetição (laços)

38. Escreva uma subrotina que imprima os números de 1 a 10 em ordem crescente.

39. Escreva uma subrotina que imprima os números de 1 a 10 em ordem decrescente.

40. Escreva uma subrotina que imprima a soma de todos os números naturais até 1000 que são múltiplos de 9.

41. Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, tem-se 3, 5, 6 e 9, cuja soma é 23. Encontre a soma de todos os múltiplos de 3 ou 5 abaixo de 1000.

42. Para que a divisão entre 2 números possa ser realizada, o divisor não pode ser nulo (zero). Escreva um programa para ler 2 valores e imprimir o resultado da divisão do primeiro pelo segundo. OBS: O programa deve validar a leitura do segundo valor (que não deve ser nulo). Enquanto for fornecido um valor nulo a leitura deve ser repetida.

43. Desenvolva um gerador de tabuada, capaz de gerar a tabuada de qualquer número inteiro entre 1 e 10. O usuário deve informar de qual número (n) ele deseja ver a tabuada. A saída deve ser conforme o exemplo a seguir ($n = 5$):

	A	B	C
1	5x1=	5	
2	5x2=	10	
3	5x3=	15	
4	5x4=	20	
5	5x5=	25	
6	5x6=	30	
7	5x7=	35	
8	5x8=	40	
9	5x9=	45	
10	5x10=	50	
11			

44. Faça um programa que peça a idade para n pessoas, ao final o programa deverá verificar se a média de idade da turma varia entre 0 e 25, 26 e 60 e maior que 60; e então, dizer se a turma é jovem, adulta ou idosa, conforme a média calculada.

45. Faça um programa que leia uma quantidade indeterminada de números positivos e conte quantos deles estão nos seguintes intervalos: [0 a 25], [26 a 50], [51 a 75] e [76 a 100]. A entrada de dados deverá terminar quando for lido um número negativo.

46. Tentando descobrir se um dado era viciado, uma pessoa o lançou *n* vezes. Dados os *n* resultados dos lançamentos, determinar o número de ocorrências de cada face.

47. Escreva um programa que leia o primeiro nome e a altura das moças inscritas em um concurso de beleza. Quando for informada a palavra **FIM** para o nome da moça o programa deverá ser encerrado e imprimir: o nome e a altura da moça mais alta e o número de moças no concurso. Considere que todas as moças têm alturas diferentes.

48. Faça um programa que leia um número indeterminado de idades. A última idade lida, que não entrará nos cálculos, deverá ser igual a zero. Ao final programa deverá escrever quantas idades foram lidas, calcular e escrever a média de idade desse grupo de idades.

49. Dado um país A, com 5000000 habitantes e uma taxa de natalidade de 3% ao ano, e um país B com 7000000 habitantes e uma taxa de natalidade de 2% ao ano, escrever um algoritmo que seja capaz de calcular iterativamente e no fim imprimir o tempo necessário para que a população do país A ultrapasse a população do país B.

50. Dado um número natural *n,* determine o número harmônico H_n definido por:

$$H_n = \sum_{k=1}^{n} \frac{1}{k}$$

51. Dado um inteiro positivo *n*, calcular e imprimir o valor da seguinte soma:

$$\frac{1}{n} + \frac{2}{n-1} + \frac{3}{n-2} + \cdots + \frac{n}{1}$$

52. Faça uma função *arctan* que recebe um número real $x \in [0,1]$ e devolve uma aproximação do arco tangente de x (em radianos) por meio da série:

$$arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

incluindo todos os termos da série até $\left|\frac{x^k}{k}\right| < 0{,}0001$.

53. Faça uma função que receba um número inteiro positivo e retorne se esse número é ou não primo. Um número inteiro é primo se ele é maior do que 1 e seus únicos divisores são ele mesmo e 1.

54. Escreva uma subrotina na qual devem ser informados dois números inteiros, um inicial e um final e, nela, calcular e mostrar a quantidade de números primos contidos no intervalo fechado entre esses números.

55. Dados *n* números inteiros positivos, calcular a soma dos que não são primos.

56. Faça um programa que calcule e escreva o valor de S:

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \cdots + \frac{99}{50}$$

57. Escreva uma função que, dado um *n* inteiro como parâmetro, retorne o fatorial de *n*.

58. Escreva uma subrotina, que dado um número natural *n*, retorne o *n*-ésimo elemento da sequência de Catalan. O *n*-ésimo termo da sequência é dado pela equação:

$$Cn = \frac{(2n)!}{(n+1)! * n!}$$

59. Um capital C, aplicado a um juro mensal i, retorna após n meses, um valor total M, denominado montante, de acordo com a seguinte expressão: $M = C \times (1+i)^n$. Escreva uma subrotina que apresente a menor quantidade de meses necessária para que se alcance o valor desejado. Os parâmetros da função devem ser o capital investido, o juro aplicado e o montante desejado.
Exemplo:

Dados:

Capital (C)= R$3.000,00;
Juro (i) = 3%;
Montante (M) = R$3.500,00.

Resultado :

Meses necessários (n) = 6.

60. Verifique que vale a igualdade matemática:

$$cos(x+y) = cos(x) * cos(y) - sen(x) * sen(y)$$

Onde x e y são ângulos em radianos. Para os cálculos de seno e cosseno, use as aproximações:

$$sen(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

$$cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

Incluindo todos os termos da série até que $\left|\frac{x^k}{k!}\right| < 0{,}0001$.

61. Escreva uma função em que o usuário entra com um vetor v, um índice k qualquer desse vetor e um número p qualquer. Essa função deve retornar o resultado L:

$$L = \frac{\prod_{\substack{j=1 \\ j \neq k}}^{n} (p - v_j)}{\prod_{\substack{j=1 \\ j \neq k}}^{n} (v_k - v_j)}$$

62. Faça um programa que leia 10 números inteiros positivos, calcule e imprima os que são números perfeitos. Um número é perfeito quando a soma de seus divisores, exceto ele próprio, é igual ao número. Por exemplo 6 é perfeito pois é a soma de 1, 2 e 3.

14.4 Exercícios de vetores

Obs: Qualquer subrotina cujo resultado seja um vetor que será usado em outro programa, deve ter esse vetor passado como parâmetro nessa subrotina.

63. Escreva uma subrotina que lê um vetor que está escrito de forma sequencial horizontalmente no Excel. Os parâmetros de entrada são: o vetor a ser criado e retornado (para depois ser usado em outras subrotinas), o número da linha e o número da coluna onde o primeiro elemento desse vetor se encontra no Excel.
Sub LeVetor(v() As Single, nl As Integer, nc As Integer)

64. Escreva uma subrotina que dados um vetor (pode usar a subrotina *LeVetor*) e um número *n,* fazer a contagem de ocorrência de *n* no vetor.

65. Ler um vetor de 6 elementos contendo o gabarito da Mega Sena. A seguir, ler um vetor de 10 elementos contendo uma aposta. Escrever quantos pontos fez o apostador.

66. Dados dois vetores de tamanho *n*, faça uma subrotina que diga se eles são iguais ou diferentes.

67. Escreva uma subrotina que mostre um vetor de forma sequencial horizontalmente no Excel. Os parâmetros de entrada são: o vetor a ser impresso no Excel, o número da linha e o número da coluna onde o primeiro elemento desse vetor deve ser escrito no Excel.
Sub MostraVetor(v() As Single, nl As Integer, nc As Integer)

68. Faça um programa que leia os elementos de um vetor por *InputBox*. A leitura dos elementos deve ser interrompida quando o valor 0 for digitado. Imprimir esse vetor no Excel com a subrotina *MostraVetor*.

69. Crie uma subrotina que dado um número inteiro *n*, criar um vetor de tamanho *n* onde nos índices pares deverá conter o valor booleano *True* e nos índices ímpares o valor booleano *False*.

70. Escreva uma subrotina que dados dois vetores (v1 e v2), crie o vetor v3 que contenha a soma de v1 com v2.

71. Escreva uma subrotina que dados dois vetores (v1 e v2), crie o vetor v3 que contenha a subtração do primeiro vetor pelo segundo.

72. Escreva uma função que tenha como parâmetros de entrada dois vetores e como saída o produto interno (produto escalar) entre esses dois vetores.

73. Escreva uma subrotina que dado um vetor, dobre o tamanho desse vetor (use o mesmo vetor) e coloque nas posições vazias os valores do vetor original multiplicados por 5.

74. Faça um programa que dados dois vetores de mesmo tamanho, gere um terceiro vetor, cujos valores deverão ser compostos pelos elementos intercalados dos dois outros vetores.

75. Crie uma subrotina que dado um vetor de qualquer tamanho, imprima dois novos vetores da seguinte forma: o primeiro vetor será composto por todos os números pares do vetor original e o segundo vetor será composto por todos os números ímpares do vetor original (considere o zero como sendo um número par).

76. Crie uma subrotina que dado um vetor de qualquer tamanho, imprima um novo vetor contendo os índices onde existe o número 0 no vetor informado.

77. Crie uma função que retorne o maior elemento de um vetor passado como parâmetro da função.

78. Crie uma função que retorne o menor elemento de um vetor passado como parâmetro da função.

79. Faça uma função que receba um vetor como parâmetro. Esta função deve retornar *True* se há elementos repetidos nesse vetor ou *False* se não há repetições no vetor.

80. Utilize a subrotina *LeVetor* para ler um vetor que está digitado no Excel. Criar uma subrotina que imprima esse vetor lido de forma diagonal nas células do Excel.
Exemplo:

	A	B	C	D	E
1	4	-1	0,9	8	
2					

4			
	-1		
		0,9	
			8

81. Dados um número real a_1, uma razão r (real) e um número inteiro *n*, alocar em um vetor os *n* primeiros termos de uma progressão aritmética.

$$PA = a_1, \quad (a_1 + r), \quad (a_1 + 2r), \quad (a_1 + 3r), \quad \dots \ [a_1 + (n-1)r]$$

82. Dados dois números reais a_1 e a_2, e um número inteiro *n*, alocar em um vetor os *n* primeiros termos de uma progressão geométrica.

$$PG = a_1, \quad a_1 q, \quad a_1 q^2, \quad a_1 q^3, \quad \dots \ a_1 q^{n-1}$$

onde $q = \frac{a_2}{a_1}$.

83. Escreva uma subrotina que, dado um vetor, inverta os elementos desse vetor. Faça a inversão no próprio vetor, ou seja, não crie um vetor auxiliar.

Exemplo: $v = [\ 5\ 6\ 8\ 14]$ então o resultado será $v = [\ 14\ 8\ 6\ 5]$.

84. Sabe-se que o comprimento de um vetor no plano é dado por $\sqrt{x^2 + y^2}$. Em um espaço *n*-dimensional a fórmula é generalizada para: $\sqrt{\sum_{i=1}^{n} x_i^2}$. Faça uma função que entre com um vetor e a saída seja o comprimento desse vetor.

85. O CPF é formado por 11 números, e os 2 últimos números são os dígitos verificadores. Apenas com os 9 primeiros números é possível descobrir quais são os dígitos verificadores de um CPF.
Vamos utilizar um exemplo para facilitar o entendimento. Suponha que os 9 primeiros números do CPF de uma pessoa são 357.240.331. Vamos descobrir quais são os dois últimos números desse CPF.

- Descoberta do primeiro dígito verificador:

  Passo1: Multiplique cada número do CPF pela sequência 10, 9, 8,...,2.

  3×10=30
  5×9=45
  7×8=56
  2×7=14
  4×6=24
  0×5=0
  3×4=12
  3×3=9
  1×2 =2

  Passo 2: Some todos os resultados das multiplicações efetuadas, isto é,
  soma = 30+45+56+14+24+0+12+9+2=192
  Passo 3: Dividir a soma (192) por 11: 192/11 =17 (sobra 5)
  Subtraia de 11 a sobra da divisão, isto é, 11-5 =6
  Passo 4: Se o resultado da subtração for maior que 9, o primeiro dígito verificador é 0; caso contrário, o dígito verificador é o resultado da subtração. Neste exemplo, o primeiro dígito verificador é 6.

- Descoberta do segundo dígito verificador:

  Passo 5: Acrescente ao CPF o primeiro dígito verificador já descoberto e multiplique cada número do CPF pela sequência 11,10,9,...2.

  3×11=33
  5×10=50
  7×9=63
  2×8=16
  4×7=28
  0×6=0
  3×5=15
  3×4=12
  1×3 =3

6×2 =12

Passo 6: Some todos os resultados das multiplicações efetuadas, isto é,

soma = 33+50+63+16+28+0+15+12+3+12=232

Passo 7: Dividir a soma (236) por 11: 232/11 =21 (sobra 1)

Subtraia de 11 a sobra da divisão, isto é, 11-1 =10

Passo 8: Se o resultado da subtração for maior que 9, o segundo dígito verificador é 0; caso contrário, o dígito verificador é o resultado da subtração. Neste exemplo, o segundo dígito verificador é 0.

Faça uma subrotina que dado um vetor (CPF com 9 números), descubra o CPF completo.

86. Execute no papel o seguinte programa, mostrando o conteúdo do vetor v1:

```
Option Base 1
Sub executar(x() As Single, y() As Single, z() As Single, resposta()
As Single)
    Dim n As Integer
    Dim i As Integer
    n = UBound(x)
    ReDim resposta(n)
    For i = 1 To n
        resposta(i) = x(i) ^ 2 - y(i) * 3 - z(i) / 2
    Next
    For i = 2 To n
        resposta(i) = resposta(i - 1) – z(i)
    Next
End Sub

Sub principal()
    Dim v1() As Single
    Dim v2(4) As Single
    Dim v3(4) As Single
    Dim v4(4) As Single
    v2(1) = 2
    v2(2) = 0
    v2(3) = -4
    v2(4) = 7
    v3(1) = 1.5
    v3(2) = -1
    v3(3) = 5
    v3(4) = 2
    v4(1) = -0.5
    v4(2) = 2
    v4(3) = -10
    v4(4) = 4
    executar v2, v3, v4, v1
End Sub
```

87. A sequência de Fibonacci é: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,...

a) Escreva uma subrotina que dado valor n natural, criar um vetor com os n primeiros números da sequência de Fibonacci.

b) Crie uma função que retorne a soma dos elementos desse vetor, ou seja, a soma dos n primeiros números da sequência de Fibonacci.

88. Dada uma sequência de *n* números reais, determinar os números que compõem a sequência e o número de vezes que cada um deles ocorre na mesma.
Exemplo: *n* = 8
Sequência: -1.7, 3.0, 0.0, 1.5, 0.0, -1.7, 2.3, -1,7
Saída: -1.7 ocorre 3 vezes
3.0 ocorre 1 vez
0.0 ocorre 2 vezes
1.5 ocorre 1 vez
2.3 ocorre 1 vez

89. Existia uma brincadeira de criptografia de textos chamada ZENIT POLAR. Funciona assim:
Z E N I T
P O L A R
Em uma palavra, por exemplo, REPOLHO, se o R está na segunda linha troca-se pelo correspondente da primeira linha (T), o E está na primeira linha, troca-se pelo correspondente da segunda linha (O), e assim sucessivamente.
Neste exemplo, a palavra REPOLHO criptografada ficaria TOZENHE.
Ao aplicar-se novamente o algoritmo sobre esta palavra retorna-se a palavra original.
Faça um programa que criptografe um texto por duas chaves dadas (neste exemplo as chaves são ZENIT e POLAR). Estas chaves podem ser modificadas

90. Chama-se *sequência de Farey relativa a* n, a sequência das frações racionais irredutíveis, dispostas em ordem crescente, com denominadores positivos e não maiores que *n*.

Exemplo: Se $n = 5$, os termos α da sequência de Farey, tais que $0 \leq \alpha \leq 1$ são:

$$\frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1}.$$

Para gerarmos os termos α de uma sequência de Farey tais que $0 \leq \alpha \leq 1$, podemos usar o seguinte processo. Começamos com as frações

$$\frac{0}{1} \quad e \quad \frac{1}{1},$$

e entre cada duas frações consecutivas

$$\frac{i}{j} \quad e \quad \frac{k}{m}$$

introduzimos a fração:

$$\frac{i+k}{j+m}$$

e assim sucessivamente enquanto $j + m \leq n$. Quando não for mais possível introduzir novas frações teremos gerado todos os termos α da sequência de Farey relativa a n, tais que $0 \leq \alpha \leq 1$.
Usando o processo descrito, determine os termos α, $0 \leq \alpha \leq 1$, da sequência de Farey relativa a n, α inteiro positivo.
Sugestão: Gere os numeradores e os denominadores em dois vetores.

91. Dada uma sequência de *n* números inteiros, determinar um segmento de soma máxima.
Exemplo: Na sequência 5, 2, -2, -7, 3, 14, 10, -3, 9, -6, 4, 1 , a soma do segmento é 33.

92. Escreva uma subrotina que, dado um vetor, ordene os elementos desse vetor. Faça a ordenação no próprio vetor, ou seja, não crie um vetor auxiliar.
Exemplo: $v = [\ 50\ 6\ 8\ 14]$ então o resultado será $v = [6\ 8\ 14\ 50\ ]$.

14.5 Exercícios de matrizes

Obs: Qualquer subrotina cujo resultado seja uma matriz que será usada em outro programa, deve ter essa matriz passada como parâmetro dessa subrotina.

93. Faça uma subrotina que lê uma matriz de qualquer tamanho do Excel, em que são fornecidos: a matriz a ser criada no VBA, o número da linha e o número da coluna onde o primeiro elemento dessa matriz se encontra.
Sub LeMatriz(M() As Single, nl As Integer, nc As Integer)

94. Faça uma subrotina que imprime uma matriz de qualquer tamanho do Excel, em que são fornecidos: a matriz a ser impressa no Excel, o número da linha e o número da coluna onde o primeiro elemento dessa matriz deve ser impresso.
Sub MostraMatriz(M() As Single, nl As Integer, nc As Integer)

95. Escreva uma subrotina que retorne a transposta de uma matriz informada.

96. Escreva uma subrotina que dada uma matriz e um número *n*, conte e escreva quantos valores maiores ou iguais a *n* ela tem.

97. Crie um programa que recebe uma matriz de inteiros positivos e substitui seus elementos de valor ímpar por -1 e os pares por +1.

98. Escreva uma subrotina que receba como parâmetro de entrada: uma matriz e dois números inteiros *l1* e *l2* menores ou iguais ao número de linhas da matriz de entrada. Troque as linhas *l1* e *l2* de posição.

99. Escreva uma função que retorne a soma dos elementos da diagonal principal de uma matriz.

100. Escreva uma função que receba como parâmetro de entrada uma matriz. Esta função deve retornar a quantidade de 1 que essa matriz contém.

101. Dada uma matriz A_{mxn}, escreva uma subrotina que imprima o número de linhas e o número de colunas nulas da matriz.

102. Escreva uma função que retorne todos os elementos de uma matriz informada concatenados em uma única variável do tipo string.

Exemlpo: $\begin{bmatrix} 5 & -1 \\ 0 & 3.5 \end{bmatrix}$ Resposta: "5-103.5"

103. Execute no papel a seguinte subrotina:

```
Option Base 1
Sub ModificaMatriz(M() As Single)
    Dim i As Integer
    Dim j As Integer
    For i = 1 To UBound(M, 1)
        For j = 1 To UBound(M, 2) - 1
            M(i, j) = M(i, j) \ 2 + M(i, j + 1) ^ 2
        Next
    Next
End Sub

Sub chama_ModificaMatriz()
    Dim i As Integer, j As Integer
    Dim Matriz(3, 3) As Single
    Matriz(1, 1) = 4
    Matriz(1, 2) = 3
    Matriz(1, 3) = 2
    Matriz(2, 1) = 1
    Matriz(2, 2) = 7
    Matriz(2, 3) = 8
    Matriz(3, 1) = 4
    Matriz(3, 2) = 2
    Matriz(3, 3) = 0
    ModificaMatriz Matriz
    For i = 1 To 3
        For j = 1 To 3
            Cells(i, j) = Matriz(i, j)
        Next
    Next
End Sub
```

104. Escreva uma subrotina que dada uma matriz quadrada, crie duas matrizes: uma matriz triangular inferior e uma triangular superior, formadas com os próprios elementos da matriz informada.

105. Verifique se todos os elementos de uma matriz que é passada como parâmetro de uma função são inteiros, positivos e apresentam raiz quadrada natural. Retorne *True* se as condições forem atendidas e *False* caso contrário.

106. Escreva uma subrotina que, a partir de uma matriz de qualquer tamanho, crie um vetor booleano com *True* para as linhas da matriz que forem nulas e *False* para as que não forem.

Exemplo:

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 3 & 1 & 0 \end{pmatrix} \qquad v = \begin{pmatrix} False \\ True \\ False \end{pmatrix}$$

107. Escreva uma subrotina dada uma matriz de qualquer tamanho, deverá criar um vetor com quatro posições contendo os quatro elementos do "canto" da matriz informada.

Exemplo: $\begin{bmatrix} 5 & 1.6 & 4 \\ 7 & 3.5 & -5 \end{bmatrix}$ Resposta: $[5 \quad 4 \quad 7 \quad -5]$

108. Escreva uma subrotina que receba uma matriz de qualquer tamanho como parâmetro de entrada. Essa subrotina deverá retornar um vetor de duas posições contendo o menor e o maior elemento da matriz informada, respectivamente.

109. Escreva uma subrotina que dada uma matriz de qualquer tamanho, deverá trocar de posições o menor e o maior elemento da matriz informada.

110. Escreva uma subrotina dada uma matriz quadrada de qualquer tamanho, deverá retornar a mesma matriz modificada da seguinte forma: multiplique cada linha pelo elemento da diagonal principal daquela linha. A diagonal principal não deverá ser alterada.

111. Escreva uma subrotina que dada uma matriz quadrada de qualquer tamanho, troque de posição a diagonal principal com a diagonal secundária. Faça as modificações na própria matriz.

Exemplo: $M = \begin{bmatrix} 2 & 4 & -1 \\ 5 & 4 & 7 \\ 12 & -8 & 3 \end{bmatrix}$ Resposta: $M = \begin{bmatrix} -1 & 4 & 2 \\ 5 & 4 & 7 \\ 3 & -8 & 12 \end{bmatrix}$

112. Escreva uma subrotina que dada uma matriz de qualquer tamanho, criar um vetor contendo os elementos da matriz cuja soma dos índices é par.

Exemplo: $M = \begin{bmatrix} 13 & 6 & -1 \\ 5 & 4 & 7 \\ 12 & -8 & 3 \end{bmatrix}$

O elemento 13 (índice (1,1)) entra no vetor, pois 1+1=2 (que é par); já o elemento 6 (índice (1,2)) não entra no vetor, pois 1+2=3 (que é ímpar).

Resposta: $v = [13 \ -1 \ \ 4 \ \ 12 \ \ 3]$

113. Os elementos a_{ij} de uma matriz $A_{n\times n}$ representam os custos de transporte da cidade *i* para a cidade j. Dada a matriz A (de qualquer tamanho) e um vetor v de itinerários (de qualquer tamanho) como parâmetros de entrada, escrever uma função para calcular o custo total da viagem.

Exemplo: $A = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 5 & 0 & 1 & 4 \\ 2 & 1 & 0 & 8 \\ 7 & 1 & 2 & 0 \end{bmatrix}$ e $v = [1 \quad 4 \quad 2 \quad 4 \quad 4 \quad 3 \quad 2 \quad 1]$

$$custo = a_{14} + a_{42} + a_{24} + a_{44} + a_{43} + a_{32} + a_{21} = 3 + 1 + 4 + 0 + 2 + 1 + 5 = 16$$

114. As matrizes A e B são quadradas e têm as mesmas dimensões. Escreva uma subrotina para que, nos elementos de A em que o índice da linha (i) for diferente do índice da coluna (j), some-se a este elemento $A(i,j)$ dois valores x e y, sendo x a soma de todos os valores da linha i da matriz B e y a soma de todos os valores da coluna j da matriz B.

115. Escreva uma subrotina que dada uma matriz de strings 8x8 representando um tabuleiro de xadrez, calcular o valor total das peças do jogo. Espaços vazios do tabuleiro são codificados como casas vazias e têm valor 0. O valor das demais peças é dado de acordo com a tabela:

Peça	peão	cavalo	bispo	torre	rainha	rei
Valor	1	3	3	5	10	50

116. Crie uma subrotina que recebe uma matriz de qualquer tamanho e tenha como saída um vetor. Essa subrotina deve transformar a matriz em um vetor. A ordem deve ser coluna por coluna.

Exemplo: $M = \begin{bmatrix} 1 & 3 & 5 \\ 15 & 7 & 8 \\ 9 & 13 & 0 \end{bmatrix}$ Resposta: $v = (1\ 15\ 9\ 3\ 7\ 13\ 5\ 8\ 0)$.

117. Implemente uma subrotina que recebe como parâmetro duas matrizes (M1 e M2) de dimensão qualquer, e tenha como saída uma nova matriz (M3) correspondente à soma das matrizes recebidas. A soma de matrizes só pode ser realizada se suas dimensões forem iguais, caso não sejam, a matriz M3 devolvida deve ser a matriz identidade com dimensão igual ao número de linhas da matriz M1.

118. Crie uma subrotina que dada uma matriz de qualquer tamanho, contar quantos elementos da matriz são múltiplos de 3, e deve fazer a soma dos que não são múltiplos.

Exemplo: $M = \begin{bmatrix} 1 & 3 & 4 \\ 12 & 14 & 7 \\ 10 & 5 & 2 \end{bmatrix}$ Resposta: Dois múltiplos e soma igual a 43.

119. Dizemos que uma matriz quadrada é um *quadrado mágico* se a soma dos elementos de cada linha, a soma dos elementos de cada coluna e a soma dos elementos das diagonais principal e secundária são todas iguais.

Exemplo: $\begin{bmatrix} 8 & 0 & 7 \\ 4 & 5 & 6 \\ 3 & 10 & 2 \end{bmatrix}$ é um quadrado mágico.

Escreva uma função em VBA que retorne *True* se uma matriz passada como parâmetro é quadrado mágico ou retorne *False* caso contrário.

120. Dizemos que uma matriz A_{nxn} é um *quadrado latino* de ordem *n* se em cada linha e em cada coluna aparecem todos os inteiros 1,2,3,...,n (ou seja, cada linha e coluna é permutação dos inteiros 1,2,...,n).

Exemplo: $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \end{bmatrix}$ é um *quadrado latino* de ordem 4.

a) Escreva uma função que recebe como parâmetros um inteiro *n*, um vetor *v* com *n* inteiros e verifica se em *v* ocorrem todos os inteiros de 1 a *n*.

b) Usando a função do item a), verifique se uma dada matriz inteira A_{nxn} é um quadrado latino de ordem *n*.

121. Uma matriz esparsa é uma matriz que tem mais de 2/3 de seus elementos iguais a zero. Faça uma função que receba uma matriz como parâmetro e retorne 1 se essa matriz é esparsa ou 0 caso contrário.

122. Dizemos que uma matriz inteira A_{nxn} é uma matriz de permutação se em cada linha e em cada coluna houver *n*-1 elementos nulos e um único elemento igual a 1.

Exemplo:
A matriz a seguir é de permutação:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Observe que

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

não é de permutação.

Dada uma matriz inteira A_{nxn}, criar uma função que verifique se A é permutação.

123. Escreva uma função que recebe duas matrizes de qualquer tamanho como parâmetro de entrada. Essa função deve retornar 1 caso a multiplicação das duas matrizes é possível e 0 caso não seja possível.

124. Escreva uma subrotina que faça a multiplicação matricial de duas matrizes informadas. Use o exercício anterior.

Para fazer este exercício, veja exemplos em: http://mtm.ufsc.br/~gatcosta/GA-Fis-e-Eng/metodo-jordan.pdf

Você pode e deve criar subrotinas e funções auxiliares. Depois, basta chamá-las corretamente.

- Crie uma função que, tendo como parâmetro de entrada uma matriz single, retorne verdadeiro ou falso caso a matriz seja ou não simétrica.

  Fuction EhSimetrica(M() As Single) as Boolean

- Crie uma função que, tendo como parâmetros uma matriz de single e um inteiro indicando uma coluna, retorne em que linha se encontra o maior valor absoluto da matriz na coluna indicada.
  Function LocalMaior(M() As single, Coluna As Integer) As Integer

- Escreva uma subrotina que crie uma matriz identidade de ordem *n.*
  Sub CriaIdentidade(M() As Single, n As Integer)

- Crie uma subrotina que tenha como parâmetros de entrada duas matrizes single para entrada de informação e dois números de linhas. O programa deverá trocar, em cada uma das matrizes, as linhas indicadas.
  Sub TrocaLinhas(M1() As Single, M2() As Single, Linha1 As Integer, Linha2 As Integer)

- Crie uma subrotina que tenha como parâmetros duas matrizes single para entrada de informação; um número inteiro, indicativo de linha; e um número K single. O programa deverá multiplicar todos os elementos da linha indicada em cada uma das matrizes pelo valor de K.
  Sub MultLinhaPorK(M1() As Single, M2() As Single, Linha As Integer, K As Single)

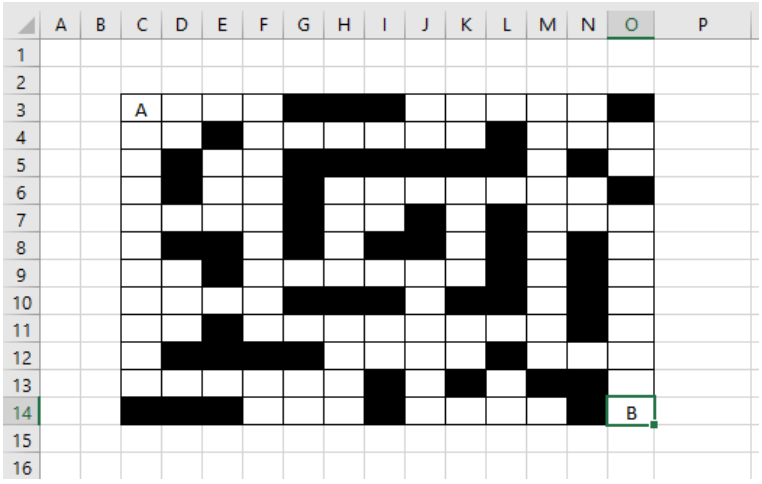
- Crie ums subrotina que tenha como parâmetros duas matrizes single para entrada de informação; dois números inteiros, indicativos de linhas; e um número K Single. O programa deverá fazer com que a primeira linha indicada seja igual a própria linha + K vezes a segunda linha Indicada.
  Sub AD(M1() As Single, M2() As Single, Linha1 As Integer, Linha2 As Integer, K As Single)

125. Desenvolver um programa que faça a decomposição LU de uma matriz dada e utilize esta decomposição para resolver um sistema de equações $Ax = b$.
Entradas: matriz A (Quadrada de ordem n) e vetor b (vetor coluna com n linhas)
Se houver incompatibilidade entre a matriz A e o vetor b, o programa deverá avisar.

Deverão ser mostradas as matrizes L e U, bem como o vetor resolução (x).

Material para consulta do método:
http://www.facom.ufu.br/~dino/disciplinas/GBC051/Decomp_LU_Lucas.pdf

126. Faça um programa que, dado um desenho como o exemplo a seguir no Excel (este desenho pode ser alterado em tamanho e as células em preto podem ser alteradas de posição e quantidade).
Resolva o problema de determinar o menor caminho saindo de A e chegando em B, sabendo-se que só é possível movimentar para as células adjacentes que sejam da cor branca. Um exemplo: da célula C6 é possível a movimentação para as células C5, C7 e D7.
O programa deve ser resolvido na memória e o caminho resposta deve ser pintado em azul.



Sugestões:

- Associe a região a uma matriz onde as células em preto estejam marcadas com -1.
- Para saber a cor de uma célula use C*ells(Linha,coluna).interior.color*, que retorna 0 para a cor preta e outros valores para as demais cores.
- Para pintar uma célula use C*ells(linha,coluna).interior.color = vbBlue*.

127. Implemente o jogo da velha, em que o tabuleiro do jogo é uma matriz no VBA de tamanho 3x3. Faça uma interface no Excel.

14.6 Exercícios de arquivos de texto e troca de planilhas

128. Crie os seguintes arquivos de texto e leia-os no VBA.

Arquivo1.txt
1.2

Arquivo2.txt
1,2

Arquivo3.txt
1;2

5.76 3.8	5,76 3,8	5;76 3;8

O que você notou?

129. Escreva uma subrotina que imprima um vetor do VBA em um arquivo de texto. Escolha se prefere que o vetor seja impresso verticalmente ou horizontalmente no arquivo de texto. Os parâmetros dessa subrotina devem ser: o vetor a ser impresso e o diretório+nome do arquivo.
Sub MostraVetorArquivo(v() As Single, arquivo As String)

130. Escreva uma subrotina que leia dados numéricos de um arquivo de texto e aloque em um vetor no VBA. Os parâmetros dessa subrotina devem ser: o vetor a ser alocado e o diretório+nome do arquivo.
LeVetorArquivo(v() As Single, arquivo As String)

131. Escreva uma subrotina que imprima uma matriz do VBA em um arquivo de texto, exatamente com a mesma estrutura que ela está no VBA. Os parâmetros dessa subrotina devem ser: a matriz a ser impressa e o diretório+nome do arquivo.
Sub MostraMatrizArquivo(M() As Single, arquivo As String)

132. Escreva uma subrotina que leia dados numéricos de um arquivo de texto e aloque em uma matriz no VBA. Os parâmetros dessa subrotina devem ser: a matriz a ser alocada e o diretório+nome do arquivo. Dica: após descobrir quantos dados o arquivo de texto contém, informar ao usuário essa quantidade e solicitar a ele que informe (pode ser por *InputBox*) as dimensões da matriz a ser criada.
LeMatrizArquivo(M() As Single, arquivo As String)

133. Utilize as subrotinas *MostraVetorArquivo, LeVetorArquivo, MostraMatrizArquivo* e *LeMatrizArquivo* para ler e imprimir vetores e matrizes de muitos exercícios desta lista, que anteriormente requeriam leituras e impressões no Excel.

134. Crie um arquivo de texto com diversos valores numéricos. Utilize a rotina *LeMatrizArquivo* para ler os dados numéricos gerados e alocar em uma matriz no VBA. Calcule a transposta desta matriz e imprima-a em um arquivo de texto utilizando a rotina *MostraMatrizArquivo.*

135. Seja a função $f(x, y) = x^2 - 3y$. Faça um programa que calcule uma matriz tal que $A(i, j) = f(j, i)$, onde as dimensões da matriz devem ser fornecidas. Imprima essa matriz em um arquivo de texto.

136. Uma subrotina recebe como parâmetro de entrada duas constantes (α e β), duas matrizes *A* e *B* de mesmo tamanho e uma letra (são 5 possibilidades de letras: a, b, c, d, e). De acordo com a letra informada, a seguinte operação deve ser realizada:

a: $A + B$ e imprimir o resultado na planilha 1
b: $A - B$ e imprimir o resultado na planilha 2
c: $\alpha(A + B)$ e imprimir o resultado na planilha 3
d: $\alpha A + \beta B$ e imprimir o resultado na planilha 4

e: $A^T + B^T$ e imprimir o resultado na planilha 5

137. Leia dois vetores que estão em arquivos separados. Verifique se eles têm o mesmo tamanho. Se sim, faça o produto interno entre esses vetores e imprima o resultado no mesmo arquivo do segundo vetor, sem perder as informações que tem nele.

138. Leia dois vetores que estão em arquivos separados. Imprima o primeiro vetor na primeira planilha do Excel e o segundo vetor na segunda planilha do Excel. Imprima na terceira planilha do Excel os dois vetores na ordem inversa.

139. Faça uma subrotina que leia um vetor de qualquer tamanho de um arquivo de texto. Em seguida, imprima esse vetor no Excel na primeira planilha, imprima na segunda planilha os elementos pares e na terceira planilha os elementos ímpares (não deixe espaço em branco na nas planilhas dois e três). Use um único laço para imprimir nas três planilhas.

Exemplo:

Primeira planilha

	A	B	C
1	2	8	
2			

Segunda planilha

	A	B	C	D
1	3	1	7	
2				

Terceira planilha

	A	B	C	D	E	F
1	2	3	1	8	7	
2						

140. Faça um programa que simule um lançamento de dados. Lance o dado 100 vezes e armazene os resultados dos lançamentos em um vetor. Imprima esse vetor em um arquivo. Depois, mostre na tela quantas vezes cada valor foi sorteado.
Dica: use um vetor de contadores (1 – 6) e uma função para gerar números aleatórios, simulando os lançamentos dos dados.

141. (Controle de cotas de disco). A XPTO Inc., uma organização com mais de 1500 funcionários, está tendo problemas de espaço em disco no seu servidor de arquivos. Para tentar resolver este problema, o administrador de rede precisa saber qual o espaço em disco ocupado pelas contas dos usuários, e identificar os usuários com maior espaço ocupado. Através de um aplicativo baixado da Internet, ele conseguiu gerar o seguinte arquivo, chamado usuarios.txt (este arquivo poderá ser modificado para teste):

```
alexandre     456123789
anderson      1245698456
antonio      123456456
carlos       91257581
cesar        987458
rosemary      789456125
```

Neste arquivo, o primeiro campo corresponde ao login do usuário e o segundo ao espaço em disco ocupado pelo seu diretório. A partir deste arquivo, você deve criar um programa que gere um relatório, chamado relatório.txt, no seguinte formato:

```
XPTO Inc.          Uso do espaço em disco pelos usuários
---------------------------------------------------------------------
Nr.  Usuário       Espaço utilizado     % do uso

1    alexandre        434,99 MB          16,85%
2    anderson        1187,99 MB          46,02%
3    antonio          117,73 MB           4,56%
4    carlos            87,03 MB           3,37%
5    cesar              0,94 MB           0,04%
6    rosemary         752,88 MB          29,16%

Espaço total ocupado: 2581,57 MB
Espaço médio ocupado: 430,26 MB
```

O arquivo de entrada deve ser lido uma única vez, e os dados armazenados em memória, caso sejam necessários, de forma a agilizar a execução do programa. A conversão do espaço ocupado em disco, de bytes para megabytes deverá ser feita através de uma função separada, que será chamada pelo programa principal. O cálculo do percentual de uso também deverá ser feito através de uma função, que será chamada pelo programa principal. As respostas solicitadas acima também deverão ser apresentadas na planilha.

Recursos adicionais: desenvolva as seguintes funcionalidades:

- Ordenar os usuários pelo percentual de espaço ocupado; (o procedimento de ordenação tem que estar separado do programa principal (Sub ou Function)
    - Mostrar apenas o n primeiros em uso, definido pelo usuário.

142. Fazer um programa que carregue informações de Nomes, Cidades e Saldo de um arquivo texto (exemplo a seguir) e os exiba na planilha Excel. Os dados deverão ser carregados na memória do programa, e deverão ser exibidos em ordem alfabética do último sobrenome, esta rotina de ordenação deve ser feita separada do programa principal. A ordenação não pode utilizar rotinas prontas do Excel.

Além disto deve ser exibido um resumo do saldo por cidade com o percentual do total da carteira.

```
Alexandre Camilo, Curitiba ,1234.56
Anderson Bartoviz, São José, 2345.66
Antonio Alvares de Moraes, Curitiba, 40231.10
Carlos Leoncio Junqueira, Ponta Grossa, 2567
Cesar  Milton Oliveira, São José, 9874.58
Rosemary Olinosky, Ponta Grossa, 78945.25
```

14.7 Exercícios de strings

143. Faça um programa que receba do usuário uma string. O programa imprime a string sem suas vogais.

144. Faca um programa que leia uma palavra (máximo de 50 letras) e some 1 no valor ASCII de cada caractere da palavra. Imprima a string resultante.

145. Faça um programa que receba uma string com os 9 primeiros dígitos de um CPF (a cada centena é colocado um ponto, portanto serão 11 caracteres) e calcule os dois últimos dígitos. Retorne o CPF no formato xxx.xxx.xxx-xx

146. Faça um programa que dado um número, verifique se ele é uma capicua, isto é, o reservo desse número é ele mesmo. Exemplos: 11, 242, 2002,...

147. Leia uma cadeia de caracteres (sem acentos) e converta todos os caracteres para maiúscula sem usar a função *UCase*.

148. Faça um programa que receba uma palavra e a imprima de trás para frente.

149. Escreva um programa que recebe do usuário uma string *S*, um caractere *C*, e uma posição *I* e devolve o índice da primeira posição da string onde foi encontrado o caractere *C*. A procura deve começar a partir da posição *I*.

150. Faça um programa que dadas duas strings, devolve o número de vezes que a segunda string está contida na primeira.

Exemplo: BANANA e ANA, resultado 2 vezes.

151. Leia uma cadeia de caracteres no formato "DD/MM/AAAA" e copie o dia, mês e ano para 3 variáveis inteiras. Antes disso, verifique se as barras estão no lugar certo, e se DD, MM e AAAA são numéricos.

152. Criar uma subrotina que separa cada elemento de uma string e os coloca em um vetor, sem usar a função *Split*.
Exemplo: String: A2c5 → Vetor (A, 2, c, 5)

153. Faça um programa de criptografia baseado no princípio da cifra de César. Use somente caracteres sem acentuação e rotação à esquerda de três posições.
https://pt.wikipedia.org/wiki/Cifra_de_C%C3%A9sar

154. Faça um programa que dado um texto (sem números e acentuação) faça a conversão para o código Morse.

155. Fazer um programa que lendo um arquivo texto qualquer, forneça:

- O número de letras no texto (espaço em branco não é letra);
    - Quantas são maiúsculas e quantas minúsculas
- O número de algarismos no texto;
- O número de espaço em branco no texto;
- O número de palavras no texto (palavras devem ter pelo menos duas letras);
- Quantas vezes aparece cada letra no texto (neste caso A=a=ã=á=à, o mesmo com as demais letras acentuadas).

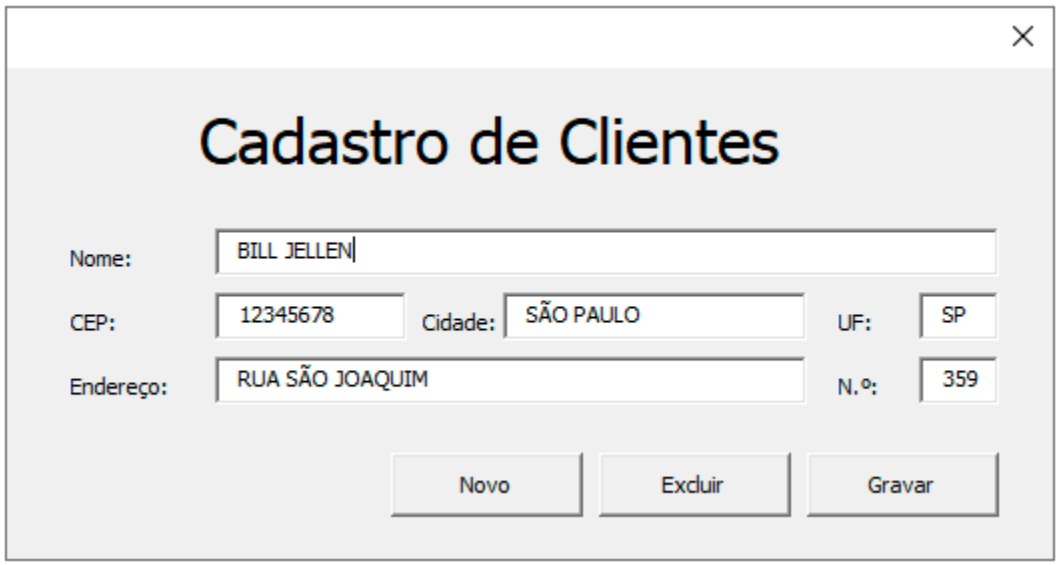
156. Faça um programa que, dada uma string, diga se ela e um palíndromo ou não. Lembrando que um palíndromo é uma palavra que tem a propriedade de poder ser lida tanto da direita para a esquerda como da esquerda para a direita. Frases também são palíndromas, onde espaços são ignorados
Exemplos:
ovo
arara
Socorram-me, subi no ônibus em Marrocos
Anotaram a data da maratona

157. Faça um programa que encontre o conjunto de 5 dígitos consecutivos na sequência a seguir que gere o maior produto. Copie e cole os dados abaixo em um arquivo de texto.

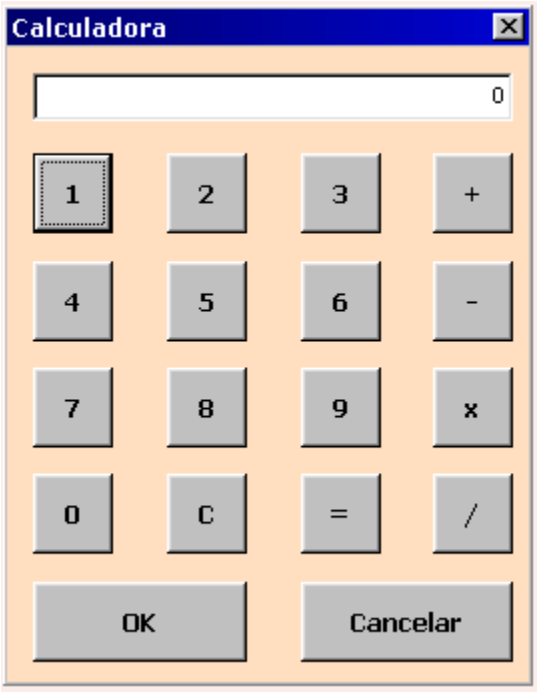
73167176531330624919225119674426574742355349194934
96983520312774506326239578318016984801869478851843
85861560789112949495459501737958331952853208805511
12540698747158523863050715693290963295227443043557
66896648950445244523161731856403098711121722383113
62229893423380308135336276614282806444486645238749
30358907296290491560440772390713810515859307960866
70172427121883998797908792274921901699720888093776
65727333001053367881220235421809751254540594752243
52584907711670556013604839586446706324415722155397
53697817977846174064955149290862569321978468622482
83972241375657056057490261407972968652414535100474
82166370484403199890008895243450658541227588666881
16427171479924442928230863465674813919123162824586
17866458359124566529476545682848912883142607690042
24219022671055626321111109370544217506941658960408
07198403850962455444362981230987879927244284909188
84580156166097919133875499200524063689912560717606
05886116467109405077541002256983155200055935729725
71636269561882670428252483600823257530420752963450

14.8 Exercícios de formulários

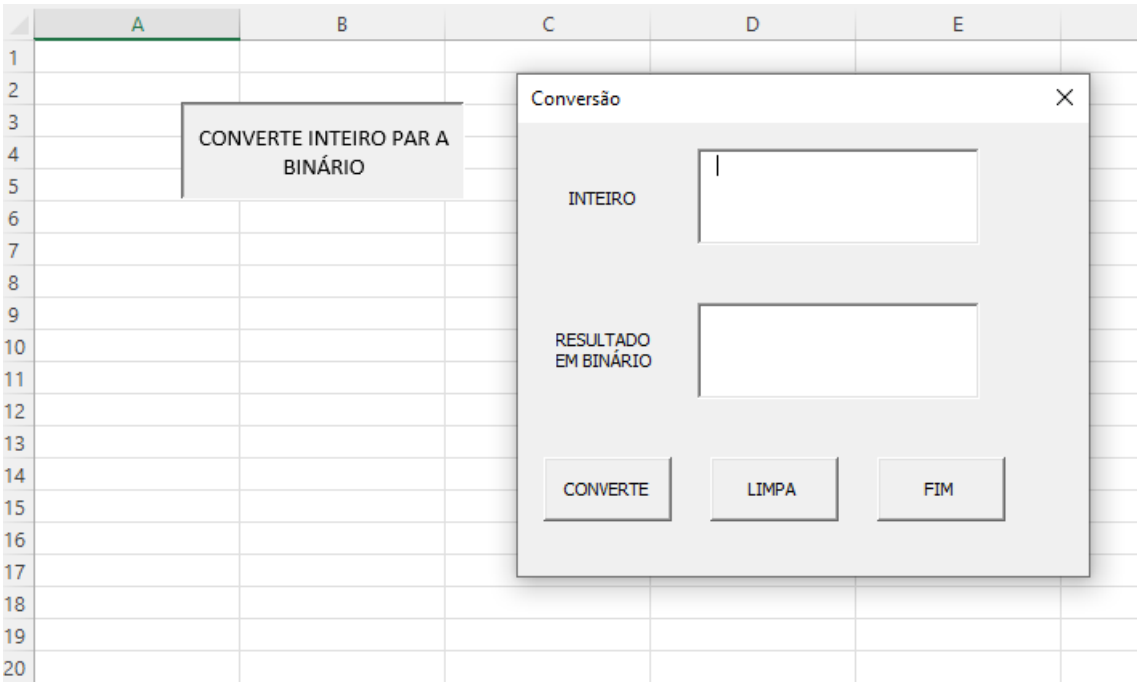
158. Criar um formulário de cadastro de clientes, onde sejam informados alguns dados (Nome, CEP, Cidade, ...) tais como na figura a seguir. Estes dados devem então ser inseridos em uma tabela (Excel) que constitui a base de clientes.



159. Implementar a seguinte calculadora usando formulário do VBA:



160. Criar um formulário para fazer a conversão de um número inteiro para binário.


CONVERTE INTEIRO PAR A BINÁRIO
Conversão
INTEIRO
RESULTADO EM BINÁRIO
CONVERTE
LIMPA
FIM